Karlsruhe Institute of Technology

Dynamic Quality Management for Cloud Labor Services

Zur Erlangung des akademischen Grades eines
Doktors der Wirtschaftswissenschaften

(Dr. rer. pol.)

von der Fakultät für Wirtschaftswissenschaften
am Karlsruher Institut für Technologie

genehmigte

DISSERTATION

von

Dipl. Phys. Robert Kern

Tag der mündlichen Prüfung:	20. Dezember 2013
Referent:	Prof. Dr. Gerhard Satzger
Koreferent:	Prof. Dr. Christof Weinhardt

Abstract

Nowadays, organizations are facing an increasing need to quickly adapt their business processes in order to react flexibly to changing markets and varying demands. This requires a scalable provisioning and "de-provisioning" of resources, especially in productive business environments. For IT resources, the huge interest in cloud computing solutions underscores this trend. However, for human resources, which are still the core of most organizational endeavors, traditional modes of operation still prohibit a flexible adaption of workforce capacity to changing needs. In this context, the notion of cloud labor services is posing a promising concept for the scalable on-demand and per-request outsourcing of human tasks to a large group of people. The success of commercial platforms like Amazon Mechanical Turk demonstrates the potential of the concept. Such platforms act as brokers between requesters who publish tasks and workers who perform the tasks in return for a monetary compensation. However, because of limited control over the individual worker contributions, ensuring an adequate quality of the work results represents a major challenge. This thesis argues that when leveraging cloud labor services in a business context, it is crucial to have an efficient and scalable quality management process in place that is capable of guaranteeing a certain well defined level of result quality. After introducing the general concept of cloud labor services in detail and providing a comprehensive overview on the existing quality management approaches, this thesis identifies a lack of approaches that meet the latter requirements to their full extent. In order to close this gap, an integrated approach for quality management of cloud labor services is developed, which combines elements of statistical quality control with a newly developed dynamic voting mechanism. The approach and a number of variations and extensions are quantitatively evaluated in a model scenario as well as in three business scenarios. As its primary contribution, this thesis represents the first application of statistical quality control to cloud labor services. On top of that, it provides the first comprehensive study on cloud labor services and specifically on quality management for cloud labor services. In doing so, this thesis does not only contribute to the academic body of knowledge but can also be considered a useful source for practitioners.

Acknowledgements

This work would not have been possible without the support and guidance of a large number of colleagues, industry partners, students and friends.

First and foremost, I would like to thank my supervisor Prof. Dr. Gerhard Satzger for his continuous encouragement and for his timely and comprehensive support in all stages of my studies. I also express my sincere gratitude to Prof. Dr. Christof Weinhardt for co-advising this thesis and providing me with constructive comments. In a similar way, I am grateful to my colleagues from the KSRI and specifically from the Service Innovation and Management group for providing an enjoyable and motivating work environment and supporting me with regular feedback that has been essential for the success of my thesis. Special thanks to Dr. Christian Zirpins who helped me focus my research in an early phase of the dissertation and accompanied my work with continuous feedback, encouragement and cooperation throughout the years. I would also like to thank Ajith Kumar Parlikad and Alexander Borek from the University of Cambridge for our successful joint research activities. Moreover, I wish to acknowledge the valuable contributions of the diploma, master and bachelor students and student assistants Hans Thies, Cordula Bauer, Pascal Wichmann, Jan Meller, David Bermbach and Sandra Rath.

The best theoretical models are worth nothing without an exhaustive validation in real live scenarios. Therefore, I particularly thank Andreas Pasing from buw Group, Fabian Dimski from bitworxx GmbH, Sacha Moufarrege, Volker Pigors, Antje Guenther, Christian Rozsenich from Humangrid GmbH, Roland Peisl, Ronald Fritz, Gerhard Pfau and Harald Kiehle from IBM and Hans Rudolf Straub as well as Maurus Duelli from Semfinder AG for the fruitful collaboration. Likewise, I want to thank all who supported me in the final stage of my work, in particular Dr. Simon Caton for his general feedback and Stefan Momma for his generous offer and commitment to proof-read several versions of my thesis.

Finally and above all, I wish to express my deepest gratitude to my dear wife Bernadett for her continuous support, her love and patience.

Contents

List of figures

List of tables

List of abbreviations

AFI	Average Fraction Inspected
AI	Artificial Intelligence
AOQ	Average Outgoing Quality
AOQL	Average Outgoing Quality Limit
API	Application Programming Interface
AQL	Acceptable Quality Level
ATI	Average Total Inspection
BPEL4People	Business Process Execution Language for People
BPM	Business Process Management
BPO	Business Process Outsourcing
CAD	Computer Aided Design
CSCW	Computer Supported Cooperative Work
CSP	Continuous Sampling Plan
CSP-1	Continuous Sampling Plan 1
DVM	Dynamic Voting Mechanism
EM	Expectation Maximization
EP	Equilibrium Probability
FN	False Negatives
FP	False Positives
GWAP	Games With A Purpose
HIT	Human Intelligence Tasks
HPS	Human Provided Service
IaaS	Infrastructure as a Service
ICD	International Classification of Diseases
ISH	IBM Insurance Service Hub
LIaaS	Labor Infrastructure as a Service
LPaaS	Labor Platform as a Service
LSaaS	Labor Solution as a Service
LTPD	Lot Tolerance Percent Defective
MLE	Maximum Likelihood Estimation
MTurk	Amazon Mechanical Turk Platform (www.mturk.com)
OC	Operating Characteristic
OCR	Optical Character Recognition

PaaS	Platform as a Service
POMDP	Partially-Observable Markov Decision Process
QM	Quality Management
RQL	Rejectable Quality Level
SaaS	Software as a Service
SLA	Service Level Agreement
SLO	Service Level Objective
SLP	Service Level Parameters
SOA	Service Oriented Architecture
SPC	Statistical Process Control
SQC	Statistical Quality Control
TN	True Negatives
TP	True Positives
VoI	Value of Information
WaaS	Workforce as a Service
WS-HumanTask	Web Service Human Task

Part I.

Foundations

1. Introduction

Driven by the advancements of information and communication technology and the acceleration of communication processes, market conditions and demands are changing more and more rapidly today. A sudden increase in demand may cause resource shortages leading to delays and deterioration in quality. In contrast, a sudden collapse in demand may result in over-capacities that can put the entire business at risk.

As a result, organizations can no longer afford to rely solely on dedicated resources. Instead, at least partially, a *scalable* provisioning and de-provisioning of resources is required. For IT resources, appropriate mechanisms are provided by cloud computing technology. However, even if the technological progress has led to a continuous reduction of manual processes, humans are still at the heart of today's organizations. For human resources, current employment models do not support a flexible adaption of workforce capacity to changing needs. When relying on internal employees, a large workforce has to be hired in order to cover peak workloads, leading to cost overheads in times of reduced demand.

Cloud labor services are a specific form of crowdsourcing that address this issue by applying the idea of cloud computing to human workforce. A coordination platform serves as an interface between requesters who need to get work done and a large crowd of workers who want to perform work in order to earn money. As it is the case in cloud computing, the service requester typically only pays for the actual efforts. The work is performed by a large and potentially distributed crowd of service workers through a Web interface in form of small, formalized tasks. An early example of such a platform is Amazon's Web marketplace Mturk[1], on which service requesters can publish open calls for so-called *Human Intelligence Tasks* (HITs). Any Internet user who meets certain skill criteria may act as a service worker and complete tasks in return for a monetary compensation while Amazon receives a percentage of the service fee for each task.

According to a 2012 study conducted by crowdsourcing.org, the annual growth of the overall crowdsourcing business accelerated from 50% in 2010 to 75% in 2011, while the participating companies account for some $375m in revenue[2] (Crowdsourcing LLC, 2012). It can be assumed that a noteworthy portion of this revenue can be attributed to cloud

[1] `http://www.mturk.com/`, last accessed on June 30, 2013.
[2] All dollar amounts referenced in this thesis represent US dollars.

labor services.[3] Turian (2012) reports that cloud labor services are ”currently deflating the traditional BPO[4] market” which is estimated by IDC to reach $202.6bn in 2016 (International Data Corporation, 2013).

On the one hand, the concept is a promising approach for addressing the increasing flexibility needs of today's businesses, on the other hand it also poses a multitude of social, legal, technical and economic challenges. A fundamental social question is whether cloud labor services at all represent a desired work model and whether the concept is capable of providing a level of payment and job security that is comparable to traditional work relationships. Legal challenges include contracting and privacy aspects while technical and economical challenges are related to the overall design and the characteristics of the marketplace and the interactions between the involved parties. However, as consistently reported in the above studies, quality management is the by far most pressing challenge (Crowdsourcing LLC, 2012; Turian, 2012). This thesis will contribute to resolving this challenge and to removing the key obstacle for the further adoption of cloud labor services.

The remainder of this chapter is structured as follows: Section 1.1 describes the problem being addressed and subsumes it into a research question. Section 1.2 then describes how the research question was approached before section 1.3 lays out the structure of the thesis. Finally, section 1.4 summarizes the development of the research work over time and refers to parts of the work that have been shared and tested with the academic community via publications, conferences and workshops.

1.1. Problem

Given that there is much less knowledge about and control over the workforce than there is in traditional work relationships, one of the most obvious challenges of the cloud labor concept concerns the quality of the work results with regard to the accuracy of the information provided by the worker. A manual validation of the work results by the requester is usually not a good choice, as it would compromise the scalability of the concept, which is supposed to be its primary advantage. Therefore, many existing quality management (QM) approaches for cloud labor services are crowd-based themselves, i.e. they outsource the QM effort back to the crowd by passing redundant tasks to the crowd or by having other workers validate or refine the original contributions. That way, they sustain the scalability of the concept, but at the same time, they may become inefficient because numerous workers may need to get involved in order to generate reliable results. A viable approach for QM of cloud labor should be both scalable and efficient. Moreover, it should be capable of meeting the specific quality needs of the requester and it should be applicable to a wide range of business scenarios. These considerations lead to the following research question which concerns the feasibility of designing a QM mechanism for cloud labor services:

Research question: *How can a scalable and efficient quality management mechanism for cloud labor services be designed in a way that it delivers results with a well defined level of quality to the requester?*

[3] About 20% of the companies who participated in the study are falling into the category of cloud labor services.
[4] Business process outsourcing

1.2. Research approach

The research question has been addressed in two steps: Starting with an in-depth literature review, the research field was first analyzed and gaps were identified. In a second step, an approach was identified for closing the gap and to this end, a novel QM approach for cloud labor was developed. In order to ensure that the new approach would meet the requirements of various use cases, those were taken into consideration already while developing it. The approach as well as variations of it were then assessed in a model scenario and in a number of case studies.

Crowdsourcing is a highly interdisciplinary research field that is being addressed by a large number of disciplines including artificial intelligence, computer science, economics, information systems, behavioral science, sociology, psychology and law. Therefore, the literature review was not limited to certain conferences and journals, but a generic review was performed based on relevant keywords and combinations of them.[5] Additionally, the publications of interdisciplinary workshops that specifically focus on crowdsourcing and human computation have been used as a starting point for a cascading literature review. An example is the annual *Human Computation Workshop* (AAAI, 2013). Also, own workshops have been conducted in 2010, 2012 (Kern et al., 2012c) and 2013 (Cuel et al., 2013). The literature was narrowed down step by step to the specific concept of cloud labor services that is being investigated in this thesis. By investigating existing approaches for QM of such cloud labor services, gaps were identified that served as a starting point for the actual contribution.

During the analysis of existing QM approaches for cloud labor services, a lack of efficient and goal-based approaches was identified which led to the formulation of the research question. For elaborating on the research question, statistical quality control was chosen as a foundation for developing a set of QM models that build the actual contribution of this thesis. These models have been evaluated using a model scenario as well as three case studies which have been performed in cooperation with a number of business partners and which demonstrated the applicability in industry practice.

The research has been performed in close collaboration with several diploma, master and bachelor students who have been supervised by the author. Results of the corresponding diploma, master and bachelor theses have been leveraged in chapters 5 and 8 (Thies, 2010), sections 6.1 and 9.1 (Wichmann, 2011), sections 6.2 and 9.2 (Bauer, 2010), as well as in sections 5.6 and 8.2 (Meller, 2012).

1.3. Structure

Figure 1.1 illustrates the structure of the thesis. It consists of four parts: *Foundations*, *model*, *evaluation* and *conclusion*.

Part I provides the foundations for the thesis and is organized as follows: This chapter 1 introduces the topic and the objectives of the thesis, defines the research question, describes the research approach and states what parts of the thesis have already been published at conferences and in journals. Chapter 2 provides the fundamentals of the concept of cloud labor services and identifies its characteristics as well as the key challenges that evolve from it. Chapter 3 investigates the existing approaches for QM of cloud labor services and identifies its limitations in order to build the rationale for the actual contribution of

[5] The keywords are *crowdsourcing*, *human computation*, *quality*, and *quality management*.

the thesis. Chapter 4 introduces relevant aspects of the field of *statistical quality control* (SQC) which are being used as a basis for the QM mechanisms developed in the thesis.

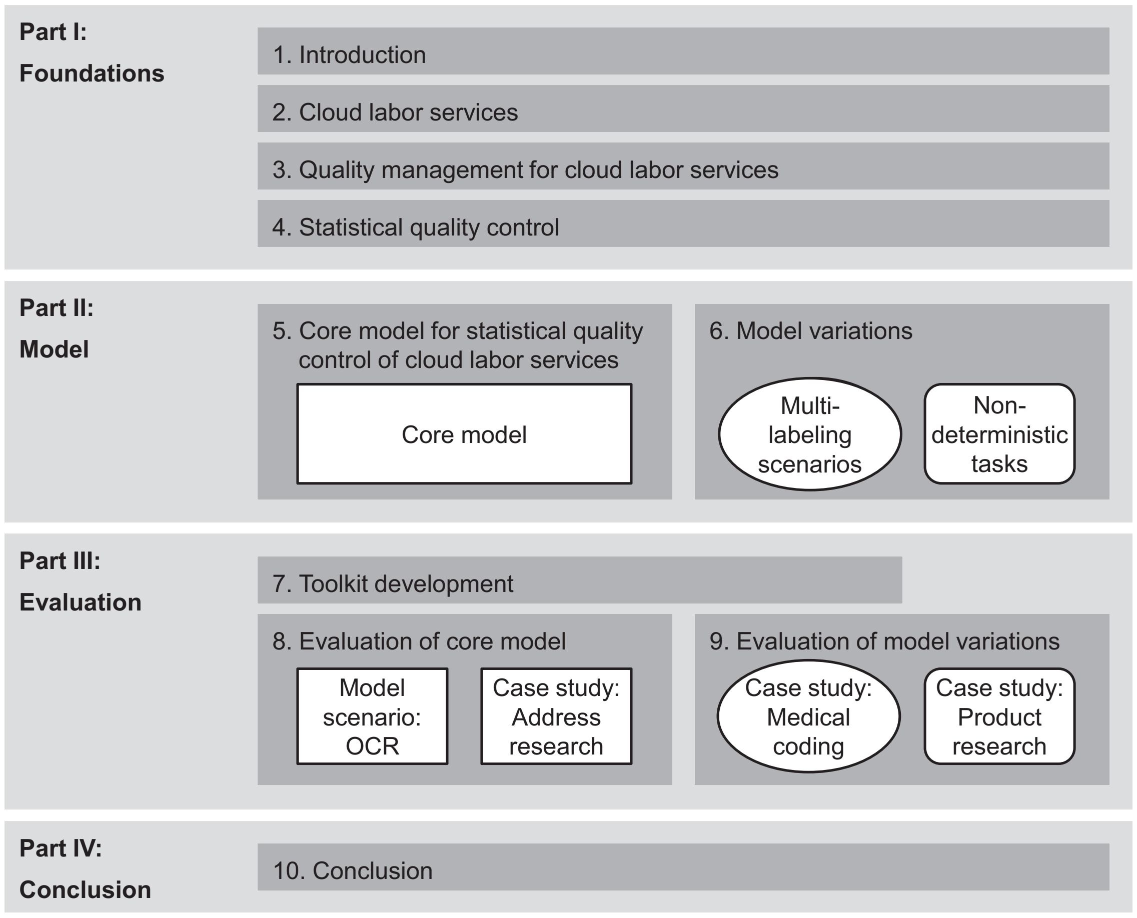

Part I:
Foundations
1. Introduction
2. Cloud labor services
3. Quality management for cloud labor services
4. Statistical quality control
Part II:
Model
5. Core model for statistical quality control of cloud labor services
Core model
6. Model variations
Multi-labeling scenarios
Non-deterministic tasks
Part III:
Evaluation
7. Toolkit development
8. Evaluation of core model
Model scenario: OCR
Case study: Address research
9. Evaluation of model variations
Case study: Medical coding
Case study: Product research
Part IV:
Conclusion
10. Conclusion

Figure 1.1.: Structure of the thesis.

The model in part II is divided into two chapters: Chapter 5 describes a generic statistical model for managing the accuracy of cloud labor services, which builds the primary contribution of the thesis. Chapter 6 complements the model development by providing variations that make it applicable to a broader set of relevant cloud labor scenarios, specifically *multi-labeling scenarios* and *non-deterministic tasks*.

The evaluation in part III consist of three chapters: Chapter 7 describes the architecture of a generic toolkit providing the foundation for evaluating the new QM mechanisms.[6] Chapter 8 then exploits the toolkit to evaluate the core model in a model scenario (*optical character recognition, OCR*) as well as in a business scenario (*address research*). Chapter 9 assesses the model variations in two additional business scenarios. The model for multi-labeling is applied to a *medical coding* scenario while the model for non-deterministic tasks is applied to a *product research* scenario.

Part IV (chapter 10) finally summarizes the contributions of the thesis and addresses its limitations as well as managerial implications.

[6] As suggested by the figure, the toolkit supports the evaluation of the model scenario and the first two case studies.

1.4. Research development

The major contributions of this thesis have already been tested and discussed with the academic community, presented at conferences and workshops and published in the respective proceedings as well as in academic journals.

General considerations about QM of cloud labor services and respective tools were presented at various academic workshops, e.g. the 1st International Workshop on Enabling Service Business Ecosystems at the 9th International Conference on Service Oriented Computing (Kern et al., 2009a), the 18th Annual Frontiers in Service Conference (Kern et al., 2009b), the 1st Enterprise Crowdsourcing Workshop at the 10th International Conference on Web Engineering (Kern et al., 2010c) and the 3rd Human Computation Workshop at the 25th Conference on Artificial Intelligence (Bermbach et al., 2011).

The primary parts of the core SQC model for cloud labor services described in chapter 5, its evaluation in chapter 8 and its fundamentals in chapter 4 have been published in the proceedings of the 8th International Conference on Service Oriented Computing (Kern et al., 2010d), the 19th European Conference on Information Systems (Kern et al., 2011), as well as in the International Journal of Cooperative Information Systems (Kern et al., 2012b) and presented at the 19th Annual Frontiers in Service Conference (Kern et al., 2010a). The model variation for non-deterministic tasks described in chapter 6 and its evaluation in section 9.2 were presented at the 16th Americas Conference on Information Systems (Kern et al., 2010e).

Parts of the case study results in section 9.1 were presented at the 16th International Conference on Information Quality (Wichmann et al., 2011), at the 20th Annual Frontiers in Service Conference (Satzger et al., 2011b), at two industry fairs, CeBIT 2011 (Kern et al., 2010b) and IBM IMPACT 2011 (Pfau & Kern, 2011) and at several practitioner events.

2. Cloud labor services

Despite all endeavors of rationalizing their business processes, organizations still face many types of tasks which cannot be fully automated but which require human intelligence or action to be completed. Cloud labor services provide human workforce on demand as a scalable service in order to meet the needs of organizations that deal with huge but varying workloads of such tasks.

After introducing the concept of cloud labor, identifying its characteristics and deducing key challenges in section 2.1, section 2.2 explores how cloud labor compares to related concepts. Section 2.3 then raises a series of challenges and general research questions. As the core contribution of the chapter, sections 2.4 to 2.6 provide an overview of the state of the art, organized into three perspectives: The perspective of the application benefiting from the cloud labor service, the perspective of the technical coordination platform and the perspective of the workforce that is represented by the people performing the actual work.

2.1. Overview

After putting the concept of cloud labor services into a historical context in section 2.1.1, section 2.1.2 formally introduces the basic concept of cloud labor in general and of cloud labor service specifically.

2.1.1. History

Cloud labor can be seen as a new form of outsourcing that has emerged driven by the advances in information and communication technology. While in general, "outsourcing is simply the farming out of services to a third party" (Overby, n.d.), cloud labor ties in with two trends regarding the granularity of what is being outsourced and whom it is outsourced to. In its first wave in the late 1980s, the concept of outsourcing was merely applied on a total business unit level, e.g. when Eastman Kodak outsourced its IT department to IBM, DEC and others (Hermes & Schwarz, 2005, p. 17). Along with advances in business process re-engineering and business process management, the granularity of the outsourced services became finer, which resulted in the concept of business process outsourcing (BPO) (Hermes & Schwarz, 2005, p. 30–31). Cloud labor services continue this development by further reducing the granularity down to ”microtasks” which represent rather atomic manual tasks with a typical execution time in the order of minutes.

A parallel development concerns the growing number of outsourcing providers being involved. While traditionally large deals with only a small number of providers were closed, the development of the Internet and the Web 2.0 era opened the doors for outsourcing tasks to large groups of individual Internet users rather than to organizations. In his 2006 Wired–article, Howe (2006a) shaped the term crowdsourcing for this phenomenon which he later defined as the ”act of a company or institution taking a function once performed by employees and outsourcing it to an undefined (and generally large) network of people in the form of an open call” (Howe, 2006b). Kleemann et al. (2008) point out that crowdsourcing takes place ”over the Internet”.

From a computing perspective, passing work to a crowd of workers has already been a common approach for dealing with complex and extensive calculations long before the invention of electronic computers (Grier, 2010). The largest computing office before the invention of the electronic computer has been the mathematical tables project started in 1938 in New York with 450 people working on mathematical calculations; the most renowned legacy of the project is the *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, which was released in 1964 and which quickly became the ”most widely circulated scientific text ever published” (Grier, 2007, p. 314–317). Originally, the term computer referred to a human being. In fact, the first automatic computers had the acronym ”AC” in their names to distinguish them from human computers (Grier, 2010). With the advances in computer technology, more and more tools have been developed that assisted humans in doing their work, as indicated by the emergence of terms like *computer aided design* (CAD) and *computer supported cooperative work* (CSCW). The degree of automation has increased step by step to a level where a large portion of business processes is fully represented and executed electronically. Of course, within those processes still a great number of tasks are performed by humans; nevertheless, to an ever-growing extent, the computer is setting the pace. This can be seen as a fundamental shift from *computer assisted humans* to *human assisted computers*.

2.1.2. Concept

Cloud labor is a specific form of crowdsourcing. The industry website Crowdsourcing.org (2012a) defines cloud labor as the act of ”Leveraging of a distributed virtual labor pool, available on-demand to fulfill a range of tasks from simple to complex. Crowdsourcing is used to connect labor demand and supply. Virtual workers perform activities that range from simple to specialized tasks”. A cloud labor platform is typically implemented as an electronic marketplace in the Internet or in an Intranet, on which a requester can submit work requests. Workers can browse work requests and choose according to their capabilities and interests what they want to work on. Depending on the business model of the platform, there may be a fixed compensation per task or a prize for submitting the best response. Commercial examples include MTurk[1] and oDesk[2].

According to Frei (2009), existing cloud labor platforms can be divided into two categories: Those that focus on well defined *tasks* and those that deal with rather complex *projects*. For the first category, there is usually a high volume of similar tasks to be performed that are issued in an automated manner. Because the execution time for those tasks is

[1] `http://www.mturk.com`, last accessed on July 1, 2013.
[2] `https://www.odesk.com`, last accessed on July 1, 2013.

usually rather short, there is naturally a relatively low payment per task. In the second category of platforms, requesters typically only submit a small number of similar work requests using a manual process. Negotiation and task execution usually require a direct interaction between the requester and the worker. As the execution effort is higher and often more specific skills are needed, payment per tasks is usually much higher than on task platforms. There is no agreed terminology yet for the two categories of cloud labor in the scientific literature. For the tasks, often the term *microtask* is used which is rather misleading because they do not necessarily have to be small. The primary difference between tasks and projects is that tasks are formally described while this is not necessarily the case for projects. Thus, tasks can be completed by the worker without inquiries to the requester. Everything that is needed to complete the task is usually clearly stated in the task description. This thesis proposes the term *programmatically managed* (or short *programmatic cloud labor*) for this type of cloud labor. In contrast, the project type of cloud labor is called *manually managed* cloud labor. The terms will be motivated in the following. Table 2.1 provides an overview of the cloud labor platform landscape.

Table 2.1.: Cloud labor landscape addressing a range from well-defined microtasks to complex projects; based on Frei (2009).

Platform type	**Platform examples**	**Task examples**
Task marketplace with programmatic interfaces (*programmatically managed* cloud labor)	▷ Amazon Mechanical Turk ▷ Crowdflower ▷ Clickworker	▷ Find email addresses or company websites ▷ Translate a product description to another language ▷ Find prices for competitive products ▷ Choose a category from a new catalog structure ▷ Write a product review ▷ Test this website and provide feedback ▷ Fill in the missing research citations in this report ▷ Build a list of universities conducting energy research
Project marketplace for hiring and managing a virtual workforce (*manually managed* cloud labor)	▷ oDesk ▷ Elance ▷ TopCoder	▷ Design a branded website ▷ Prepare an outline for a conference presentation ▷ Contact all confirmed attendees for an event ▷ Program a software module ▷ Design a new edible adhesive ▷ Develop a new security algorithm ▷ Develop an eCommerce website ▷ Inbound/outbound calls (sales, market research, support)

The general objective of cloud labor is to deliver scalable workforce to organizations that have to deal with a huge, but varying workload of manual efforts. However, scalability is not only a characteristic of the cloud labor platform, but also must be supported by the requester who must have enough bandwidth in order to publish requests, coordinate work and handle responses quickly enough. On the second category of platforms, on which projects are *manually managed* by the requester, the scalability with regard to the number and complexity of projects that can be handled is limited to the requester's bandwidth. From that point of view, the scalability characteristic is more pronounced for programmatically managed cloud labor, for which such a general limit does not exist.

This thesis focuses on programmatically managed cloud labor. However, to some extent the considerations can also be applied to manually managed cloud labor. The concept of programmatically managed cloud labor involves three roles:

1. A requester, ideally represented by an electronic business process that comprises tasks which either cannot be automated today or for which automation would be too expensive or too time consuming.
2. A large group of suitable internal or external remote workers (”people cloud”), acting as a virtual workforce, available to work on the tasks.
3. A web-based coordination platform (cloud labor platform), acting as an interface between the business process and the workers.

Figure 2.1 illustrates the basic concept of programmatically managed cloud labor. The service requester can be seen as a customer while the coordination platform is providing a service by means of the workers. The requester issues tasks as Web service calls via the coordination platform on which they can be browsed and processed by workers using a personal computer or mobile device. After completion of the task, the workers pass back responses, which are consolidated by the platform and finally returned to the requester. Even though the tasks are issued electronically, the actual work does not have to be performed with the help of a computer and, therefore, may not only exploit the cognitive or intellectual capabilities, but also physical capabilities of the worker. For example, a worker may be asked to check the condition of a specific street lamp that had been reported to be defective. A cloud labor platform may deliver any type of work or action that humans are capable to perform. As the platform has only limited control over the remote workers, one

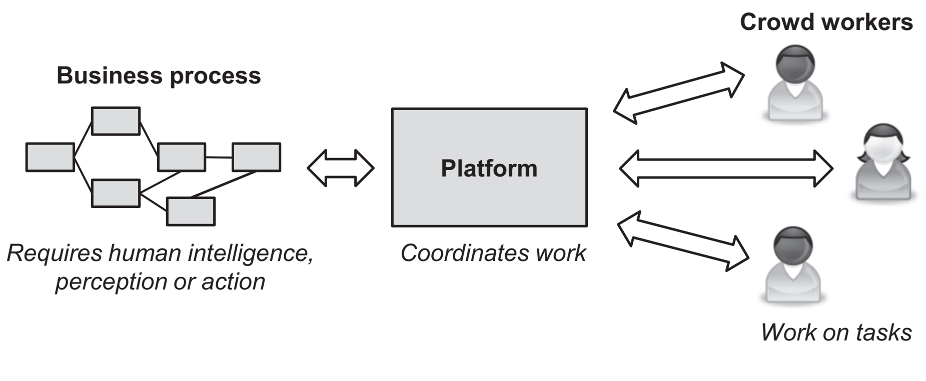


Figure 2.1.: Basic concept of programmatically managed cloud labor.

obvious challenge is quality management. From the perspective of the business process, the cloud labor platform should deliver an automated service for which a well defined level of quality is guaranteed. It does not have to be visible to the business process that the service is actually provided by humans. Therefore, programmatically managed cloud labor provides additional flexibility when designing electronic business processes and they remove the need for a general distinction between automated and manual steps.

The scalability of cloud labor is enabled by the fact that there is a large group (”crowd”) of workers available. An open call among the workers is used in order to ensure that new tasks are being promptly discovered by an adequate number of suitable workers and the requested responses are delivered in time.

Considering its service characteristic and its scalability, this thesis introduces the term *cloud labor services* for the services delivered by programmatically managed cloud labor. Cloud labor services are defined as ”Web based services that deliver human intelligence, perception, or action to customers as massively scalable resources”.

Because of the scalability characteristic, programmatically managed cloud labor additionally requires a scalable implementation of the complete task invocation cycle comprising task submission, worker allocation, progress monitoring, quality control, payment and result integration. This cycle is indicated by figure 2.2. Scalable basically means that all the steps either need to be automated or recursively implemented as cloud labor services. Manual steps performed by employees in a traditional way would compromise the scalability of the whole approach, except if there is a sufficient number of employees available to handle the steps even in times of high workload.

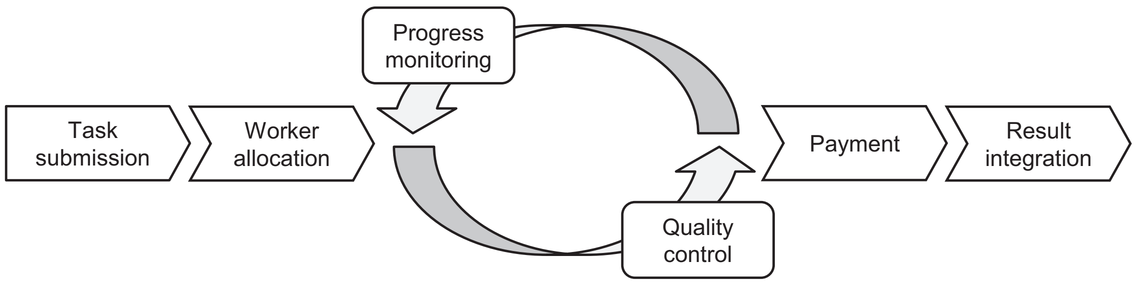


Figure 2.2.: Invocation cycle of a cloud labor service.

Obviously, there needs to be an incentive for the worker in order to spend time on cloud labor services. This incentive may be monetary as on existing cloud labor platforms like MTurk or non-monetary. An example for a non-monetary incentive are *games with a purpose* (von Ahn & Dabbish, 2008), in which the tasks are designed as games and the workers (or *players*) perform work just for fun.

2.2. Related concepts

This section describes, how the concepts of cloud labor and cloud labor services overlap with related concepts, namely crowdsourcing, human computation, social computing, collective intelligence and also business process management (BPM), service oriented architecture (SOA) and cloud computing.

While there is a direct overlap with the scope of crowdsourcing, human computation, social computing and collective intelligence that is illustrated by figure 2.3, BPM and SOA complement the concept of cloud labor by providing technical concepts for their implementation. Cloud computing shares the basic objective of cloud labor to provide resources as scalable services but focuses on IT resources rather than on human workforce.

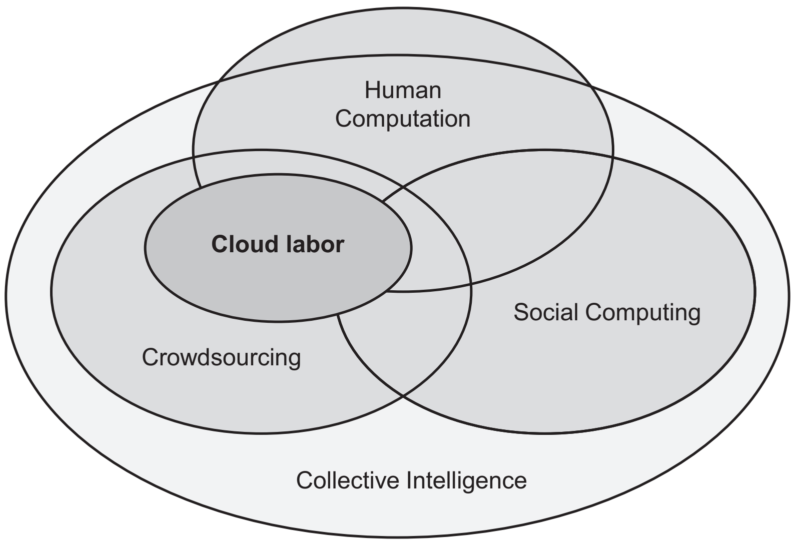


Figure 2.3.: Overlap of the cloud labor concept with similar concepts; based on Quinn & Bederson (2011).

2.2.1. Crowdsourcing

As mentioned in section 2.1.1, crowdsourcing is defined as the "act of a company or institution taking a function once performed by employees and outsourcing it to an undefined (and generally large) network of people in the form of an open call" (Howe, 2006b). Cloud labor represents a specific form of crowdsourcing. Apart from cloud labor, crowdsourcing.org (Crowdsourcing.org, 2012a) segments crowdsourcing into the following concepts:

- *Crowd creativity*: "Tapping of creative talent pools to design and develop original art, media or content. Crowdsourcing is used to tap into online communities of thousands of creatives to develop original products and concepts, including photography, advertising, film, video production, graphic design, apparel, consumer goods, and branding concepts". Prominent examples include *99 designs*[3], a platform that hosts public design contests, and *Spreadshirt*[4], a platform for the crowdsourced design of personalized apparel.

- *Crowd knowledge*: "Development of knowledge assets or information resources from a distributed pool of contributors. Crowdsourcing is used to develop, aggregate, and share knowledge and information through open Q&A, user-generated knowledge systems, news, citizen journalism, and forecasting". This segment covers traditional crowdsourcing applications like Wikipedia[5] and also the concept of "the wisdom of crowds" introduced by Surowiecki (2004). According to him, when aggregating imperfect intuitive judgements of a group of people in the right way, its collective intelligence is often excellent. An example for commercial use of the concept is Ask500People[6], a website that gathers feedback from hundreds of independent voters in minutes.

[3] `http://99designs.com/`, last accessed on July 1, 2013.
[4] `http://www.spreadshirt.com/`, last accessed on July 1, 2013.
[5] `http://www.wikipedia.org`, last accessed on July 1, 2013.
[6] `http://www.ask500people.com/`, last accessed on July 1, 2013.

- *Open innovation*: "Use of sources outside of the entity or group to generate, develop and implement ideas. In a world of widely distributed knowledge, where the boundaries between a firm and its environment have become more permeable, companies cannot afford to rely entirely on their own research and ideas to maintain a competitive advantage". The term open innovation was originally introduced by Chesbrough (2003). An example application is InnoCentive[7], a problem solving marketplace on which organizations can publish challenging problems. Any Internet user can propose solutions (Lakhani et al., 2007).

  Crowd labor differs from the other segments regarding its scope and regarding the way the quality of the work results is being managed. Crowd creativity, crowd knowledge and open innovation are represented by targeted applications implemented as web-based communities which are led by shared goals. Members of the community are contributing assets like texts, photographs, designs or ideas which are then evaluated or gradually improved by peers. The quality of an individual contribution may be questionable, but because many community members are interacting, there is an implicit quality control which sorts out inferior contributions or gradually improves them. In contrast, cloud labor is represented by all-purpose platforms that can be utilized for any type of remote work. The quality of the work results needs to be managed explicitly depending on the type of the application and according to the requester's requirements. Crowd creativity, crowd knowledge and open innovation may be applications of cloud labor, but they do not necessarily have to be. For example, Wikipedia belongs to the segment of crowd knowledge but does not represent cloud labor, whereas 99designs could be seen as both, cloud creativity and cloud labor. As indicated by figure 2.3, cloud labor can be seen as a sub-concept of crowdsourcing. Like crowdsourcing, cloud labor outsources work to a crowd of people and utilizes an open call for allocating workers to tasks.

2.2.2. Paid crowdsourcing

The term *paid crowdsourcing* shaped by Frei (2009) addresses those forms of crowdsourcing in which a monetary incentive is used. From that point of view it is orthogonal to the segments defined in the previous section. Frei defines paid crowdsourcing as "the act of outsourcing paid work of all kinds to a large, distributed group of workers using a technology intermediary that helps oversee the definition, submission, coordination, acceptance and payment for the work done". Note that this definition is actually broader than crowdsourcing because it does not necessarily require an open call. Therefore, it is rather inconsistent with the definition of crowdsourcing. As the term is only rarely used in the scientific literature, it is omitted from figure 2.3.

2.2.3. Human computation

Quinn & Bederson (2011) argue that the modern usage of the term human computation was inspired by von Ahn's 2005 dissertation of the same name. Von Ahn defines human computation as "a paradigm for utilizing human processing power to solve problems that computers cannot yet solve" (von Ahn, 2005). By taking into account definitions from a series of other papers, Quinn & Bederson (2011) come to the conclusion that there is a consensus about the following characteristics of human computation:

[7] `http://www.innocentive.com`, last accessed on July 1, 2013.

- ”The problems fit the general paradigm of computation, and as such might someday be solvable by computers.”
- ”The human participation is directed by the computational system or process.”

The authors explicitly point out that traditional crowdsourcing applications like Wikipedia are excluded because Wikipedia was designed as a ”collaborative writing project” and not to replace a machine.

It can be concluded that one major difference between human computation and cloud labor is that human computation is about dealing with *computational problems* while crowdsourcing and specifically cloud labor is dealing with *work* in general. Out of the three human capabilities covered by the definition of programmatically managed cloud labor in section 2.1.2 only ”intelligence” matches with the idea of human computation while ”perception” and ”action” do not because they do not represent computational problems. There are several types of applications which satisfy the definition of programmatically managed cloud labor but not the one of human computation: Filling in a survey or a poll can be regarded as work, but it does not represent a computational problem. Another example is any form of physical activity like asking a worker to take a picture of a specific building and send it back to the requester. Therefore human computation is narrower than cloud labor, but at the same time it is broader: While cloud labor is limited to the field of crowdsourcing, human computation comprises any scenario in which one or more humans are working on a computational problem and are directed by a computational system or process. There is neither a crowd required nor an open call. Figure 2.3 indicates the overlap between cloud labor and human computation.

Games with a purpose

For programmatic cloud labor there is another difference that concerns the incentives being used to attract the workers. While programmatically managed cloud labor always relies on a financial incentive, human computation does not necessarily do so. It explicitly comprises the concept of *human computation games* or ”games with a purpose” (GWAP) (von Ahn & Dabbish, 2008) that ”constructively channel human brainpower using computer games” (von Ahn, 2005). In this concept, people are playing a game just for fun, but implicitly they are actually performing work. Indeed, the early works on human computation have been in the area of games. The most noted one is surely the ESP game developed by von Ahn & Dabbish (2004). When Luis von Ahn originally shaped the term human computation, it was entirely focusing on unpaid contributors (von Ahn, 2005).

In order to illustrate the idea of human computation games, the ESP game is described here in more detail, following von Ahn & Dabbish’s (2004) original paper. It is played by two players over the Internet, who are randomly paired by being picked from a pool of available players. The same random picture is shown simultaneously to both players and they are asked to guess what words come into the mind of the partner player and type them in. As there is no way to communicate, the only commonality between the players which they are aware of is that they are looking at the same picture. So naturally, they type in words that describe what they see on the picture. For each match they both earn a certain number of points. The actual objective of the game is to attach meaningful labels to images, e.g. ”cat”, ”black”, ”sitting”. ESP stands for *extra sensory perception* and reflects the idea that the players have to ”think like each other” (von Ahn & Dabbish, 2004).

The concept had been licensed by Google and has been extremely successful as the *Google Image Labeler* which was available in Google Labs for many years. As of July 2008, more than 200,000 images had been labeled by some 50,000 players (von Ahn & Dabbish, 2008). In 2011 the service was discontinued when Google decided to close down the test bed Google Labs (Google Inc., 2011).

2.2.4. Social computing

Social computing applications comprise ”blogs, wikis, social bookmarking, peer-to-peer networks, open source communities, photo and video sharing communities, and online business networks” and can be characterized as ”applications and services that facilitate collective action and social interaction online with rich exchange of multimedia information and evolution of aggregate knowledge” (Parameswaran & Whinston, 2007).

According to Quinn & Bederson (2011), the ”key distinction between human computation and social computing is that social computing facilitates relatively natural human behavior that happens to be mediated by technology, whereas participation in a human computation is directed primarily by the human computation system”. The same distinction can be also made for cloud labor services because they are actively directed by a platform rather than by natural human behavior. The idea of social computing can be combined with the concept of cloud labor by building social networks of workers that share knowledge and jointly work on tasks in order to build synergies, gain better results and have more fun than when working alone.

2.2.5. Collective intelligence

While Lévy (2001) describes collective intelligence as ”intelligent communities, as open-minded, cognitive subjects capable of initiative, imagination and rapid response”, a recent definition by Malone et al. (2009) very broadly defines it as ”groups of individuals doing things collectively that seem intelligent.” As illustrated by figure 2.3, collective intelligence represents the basic principle and ”breeding ground” behind all three, crowdsourcing, social computing and human computation. The latter one is only included as far as a group of humans is involved, which is always the case for social computing and crowdsourcing. By being a specific form of crowdsourcing, cloud labor also represents a subset of collective intelligence.

2.2.6. Human tasks in business process management and SOA

Scheer & Brabänder (2010) define BPM as ”a structured approach employing methods, policies, metrics, management practices, and software tools to coordinate and continuously optimize an organization’s activities and processes”. From a technical perspective, BPM is closely related to the concept of SOA that is defined as ”a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains” (OASIS, 2006).

Within the fields of BPM and SOA, there are a series of IT concepts like *Web Service human task* (WS-HumanTask) (Agrawal et al., 2007a) and *business process execution language for people* (BPEL4People) (Agrawal et al., 2007b) that support the orchestration of human activities as components of service oriented architectures or electronic business processes. Like cloud labor services, these concepts allow for the provisioning of work through electronic interfaces. By focusing on technical interfaces, language design and protocols, they complement the idea of cloud labor services as they offer a technological framework for the implementation of human tasks.

The biggest difference lies in the variety of tasks, in the way they are assigned to people and in the workforce they are assigned to. Cloud labor service tasks are made available to a rather large and undefined crowd of people who choose from a large variety of tasks based on their capabilities and desires. In contrast, WS-HumanTask and BPEL4People uses a role based assignment of tasks to employees in an organizational environment. Depending on well defined responsibilities, tasks are assigned to specific employees or groups of employees who complete them as part of their daily business. There is typically only a small set of task types that may be processed by a specific person, i.e. there is a higher degree of specialization than for cloud labor services. Because there is more control over the workforce and there are fewer degrees of freedom, aspects like quality management, motivation and incentives play a less important role than for cloud labor services. Nevertheless, human tasks in BPM and SOA can likely profit from the considerations made in this thesis while cloud labor services can benefit from the technological framework of human tasks in BPM and SOA.

Human provided services (HPS) (Schall, 2011) go one step further than WS-HumanTask and BPEL4People by providing a framework for modeling human expertise and supporting its discovery and provisioning in service-oriented environments. HPS can be seen as a technical framework for implementing cloud labor services. Besides the definition of protocols and interfaces, HPS also provides an interaction model for structuring and delegating work.

2.2.7. Cloud computing

Cloud computing can be defined as ”a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction” (Mell & Grance, 2011).

According to Mell & Grance, the fundamental characteristics of cloud computing are:

- It provides ”on demand self-service”, i.e. there is no human interaction required with the service provider.
- The service can be used through ”broad network access” using standard computing devices like laptops or mobile phones.
- A ”resource pooling” mechanism serves multiple clients by dynamically assigning resources to them depending on the demand.
- Capabilities are provided with ”rapid elasticity”, i.e. the resources are provisioned and de-provisioned in a scalable way.
- It represents a ”measured service”, i.e. a metering capability monitors, controls and reports the resource usage per client.

There is a fundamental symmetry between cloud labor services and cloud computing. Basically, cloud labor services are for human workforce what cloud computing is for IT services. According to section 2.1.2, all the characteristics of cloud computing apply to cloud labor services as well: Cloud labor services are issued through a programmatic interface and delivered by a pool of workers who are dynamically allocated to the appropriate tasks. This results in a highly scalable service. Efforts are tracked and billed per requester. The overlap between the two concepts will be further discussed in section 2.5.2 when considering service models for cloud labor services.

2.3. Challenges

The concept of cloud labor poses a series of challenges that concern a wide variety of disciplines. In this chapter, the key challenges are identified and general research questions are deduced. Because of the overlap with the fields of crowdsourcing and human computation, the challenges can be derived from those research fields.

In order to structure requirements for crowdsourcing and human computation, Alonso (2011) proposes a three role perspective similar to the roles of the basic cloud labor service concept: The "experimenter view" represents the perspective of the requester who is designing the tasks, the "engine view" represents the technical platform and the "human view" represents the perspective of the workers. In this thesis, more generic terms are used that better match the terminology of the cloud labor services concept: Application view, platform view and workforce view.

For each of the perspectives, the subsequent sections 2.3.1 to 2.3.3 describe the key challenges along with corresponding research questions. Table 2.2 serves as a chapter overview by summarizing the challenges clustered by the three perspectives.

Table 2.2.: Challenges of the cloud labor service concept structured by the perspective of the application, the platform and the workforce.

Perspective	**Challenge**	**Source (selection)**
Application	Identification of relevant tasks	(Shahaf & Amir, 2007)
	Workflows and task granularity	(Kittur et al., 2011)
	Privacy, copyright and compliance	(Felstiner, 2011)
Platform	Deployment and service models	
	Worker to task matching and allocation	(Ambati et al., 2011; Yuen et al., 2011)
	Crowd management	(Schall, 2011)
	Quality management	(Snow et al., 2008; Kittur et al., 2013)
	Technical infrastructure and standards	(Alonso, 2011; Schall, 2011)
	Interfaces and usability	
Workforce	Motivation and incentives	(Kaufmann et al., 2011)
	Education and feedback	(Mason & Watts, 2012)
	Work model	(Felstiner, 2011; Silberman et al., 2010)

Later in chapter 2, sections 2.4 to 2.6 will again take up the three perspectives and describe to what extent the corresponding challenges have already been addressed by the state-of-the-art.

2.3.1. Application challenges

The application view addresses the identification of relevant applications and tasks, the decomposition into manageable work units and the recomposition of the work results as well as privacy, copyright and compliance considerations.

Identification of relevant tasks

For the requester, one of the initial challenges is to identify relevant tasks to which the cloud labor concept can be applied and how the cloud labor services can be integrated into existing business processes or combined with automated tasks. Often, at least a portion of the task can be automated so the question is how to define the manual portions and their interplay with the automated portion (Shahaf & Amir, 2007). Corresponding research questions are:

1. What are suitable types of tasks ”that are easy for humans and hard for computers” (Shahaf & Amir, 2007; Alonso, 2011), i.e. what tasks can be performed by humans in a better or cheaper way than by computers?
2. How can automatic tasks and manual tasks be combined in order to gain the optimal synergies? (Alonso, 2011)
3. How can cloud labor services extend existing business processes? (Vukovic et al., 2010)
4. What are suitable business cases for applying the cloud labor concept? How can it be applied to the enterprise scenarios (Alonso, 2011; Vukovic et al., 2010)?

Workflows and task granularity

Once a task has been identified, the second challenge is to adjust the granularity of the tasks to a level that can be handled by individual workers. Therefore, workflows need to be defined which decompose complex tasks into smaller ones and recompose the responses returned from the workers (Kittur et al., 2011). Relevant research questions are:

1. How can a complex task be decomposed into smaller entities that can be handled by a human in an acceptable amount of time?
2. How can the results of the individual sub-tasks be reassembled in order to gain a consolidated result?
3. What are the decision criteria for deciding whether to decompose a task or not?

Privacy, copyright and compliance

Another important challenge is to address potential privacy, copyright and compliance needs. In general, all the tasks and responses exchanged on a cloud labor service platform represent information that may be subject to data privacy protection or may represent business secrets. Even if the requesters add appropriate privacy policies to their participation agreements ”it would be naive for firms to count on those agreements, or on vendors in general, to protect intellectual property” (Felstiner, 2011). Also, data protection acts may prohibit the disclosure of specific types of information either in general or restrict the use of the data, for example with regard to the countries it may be transferred to. A third challenge is that information generated or researched by workers within the scope of a cloud labor service (e.g. in text authoring or image retrieval applications) may be subject to copyright. Corresponding research questions are:

1. How can the privacy of the requesters and their customers be guaranteed even when dealing with confidential information?

2. How can the privacy of workers be guaranteed given that the cloud labor service provider or requester might gather a lot of information about the individual workers over time?

3. How can the platform or the requesters respect the copyright of texts and multimedia data that workers research from the Internet?

2.3.2. Platform challenges

The platform view concerns the technical and economical challenges concerning the cloud labor platform including deployment models, quality management mechanisms, the design of the requester and worker interfaces, the technical architecture of the platform as well as the underlying protocols and standards. Another aspect is the matching and allocation of workers to tasks and tasks to workers.

Deployment and service models

Similar to cloud computing, various deployment and service models can be conceived for cloud labor platforms with respect to the level of service they deliver and the type of workforce they rely on. The platform may be operated by the requester or by an external company. The definition and tailoring of the tasks and the management of the result quality may be performed by the requester, by the platform owner or again by a third party service provider. The workforce may be provided by the requester, by an external organization or it may consist of freelancers or Internet users who want to earn some extra money in their spare time. There may also be a combination of platforms and workforces that are interconnected in some way. Corresponding research questions are:

1. What are viable deployment models for task market places and how do they relate to the corresponding models of cloud computing?

2. What are viable service models of cloud labor service platforms with respect to the workforce they rely on?

3. How can multiple cloud labor platforms and workforces be combined in an effective way?

Worker to task matching and allocation

One of the key benefits of cloud labor is that it provides access to a huge pool of skills, experiences, cultures, beliefs, opinions, habits and desires that makes it applicable to almost any kind of human work one can think of, assuming that it can be performed or at least initiated electronically. However, an effective worker to task matching and allocation is needed in order to ensure that requesters find the right workers for their tasks and workers find appropriate tasks to work on. Relevant research questions are:

1. How can tasks be effectively matched to workers while taking into account their individual preferences, skills, experience, strengths and weaknesses? (Law & Ahn, 2011, p. 6)

2. How can task properties and worker characteristics be formally modeled in a comprehensive way that supports matching of tasks to workers?

3. How can a taxonomy of skills be designed that captures relationships and proximity between skills?

Crowd management

Especially if an organization relies on its own workforce for delivering cloud labor services, it has to be carefully managed in order to ensure that the employees are fully utilized but not overloaded, that their expertise is leveraged and that they are happy with the work conditions. But also for marketplaces that rely on Internet users, a proper crowd management is an important instrument for establishing and maintaining a powerful and loyal workforce. Relevant research questions are:

1. What mechanisms allow for an effective workload and skill management on task platforms? (Alonso, 2011)
2. What tools can facilitate an effective communication with the crowd workers?

Quality management

Quality management can be regarded as one of the key challenges of cloud labor services. Kittur et al. (2013) report that it has "arguably received the most attention so far". Because of the anonymity of the workforce, there is only little control over the quality of the work results delivered by an individual worker (Suri et al., 2011). This becomes particularly obvious when dealing with a workforce of Internet users. Relevant research questions are:

1. How can a crowd of remote workers be coordinated in a way that satisfying work results can be achieved with minimal effort?
2. What factors are affecting the quality of work results and what are suitable quality metrics?
3. How can reputation or trust be utilized in order to predict the quality of work results delivered by a worker?
4. How can fraud be discovered or prohibited?

Technical infrastructure, protocols and standards

The concept of cloud labor is enabled by information technology and most of the characteristics of the cloud labor platform are basically represented by IT. This applies to the architecture and design of the cloud labor platform, the computing, storage and network infrastructure as well as the underlying protocols and standards. Other important aspects are the network bandwidth and the computing devices that can be expected on the worker side. Related research questions are:

1. How can a reliable and highly scalable cloud labor platform be designed?
2. How can existing protocols and standards be enhanced in order to address the specific needs of cloud labor applications? What protocols and standards allow for integration with other enterprise systems (Alonso, 2011)?
3. How can skill profiles and work descriptions be represented in a standardized way?

Interfaces and usability

The acceptance of cloud labor marketplaces will not at least depend on their interfaces and usability. This is not limited to a beautiful design of the screens but primarily concerns the requester and worker experience as a whole. Just because cloud labor is IT based, it does not mean that all activities have to be performed in front of a computer. Maybe other devices can be introduced or workers can organize into teams who even cooperate physically and just submit their results on a computer in the end. Relevant research questions are:

1. How should the interfaces be designed in order to make the interaction with the cloud labor service platform as easy, unambiguous, encouraging and efficient as possible for all involved parties? (Alonso, 2011)
2. What interfaces could support the virtual or physical cooperation of workers? (Law & Ahn, 2011, p. 6)

2.3.3. Workforce challenges

The workforce view comprises the motivation of the workers by means of appropriate incentives (e.g. payment), the importance of education and feedback, the overall acceptance of the work model as well as contracting and fiscal considerations.

Motivation and incentives

One of the basic principles that cloud labor services inherit from crowdsourcing is that work requests are published in an open call. It is up to the worker to decide what projects or tasks to work on. Thus, appropriate incentives need to be in place that attract suitable workers. Corresponding research questions are:

1. What are suitable incentives and incentive schemes that motivate workers to contribute continuously and with high quality? (Law & Ahn, 2011, p. 6)
2. What combinations of incentives are feasible and can leverage additional advantages, e.g. how can monetary incentives be mixed with non-monetary ones in order to save costs?
3. What incentive schemes can be applied to a private workforce of employees that anyway receive a more or less fixed salary? (Vukovic et al., 2010)

Education and feedback

The capabilities of the workers are obviously one of the most important prerequisites for a successful application of the cloud labor concept. This involves fundamental skills in relevant disciplines but may also comprise specific skills about the project. Relevant research questions include:

1. How can education mechanisms be designed that seamlessly integrate into the concept of cloud labor?
2. How can feedback mechanisms be designed that support a continuous improvement of the worker skills?
3. What effect do feedback mechanisms have on the quality of the work results and on the worker satisfaction?

Work model

The concept of cloud labor differs significantly from traditional employment models. On the one hand, it may lead to advantages as it may reduce the barriers to find a job and increase the flexibility regarding the work schedule and work location; on the other hand, there may also be the risk of jeopardizing traditional work models with regard to lay-off protection, minimum wages and pension funds (Felstiner, 2011). Furthermore, legal contracts and taxes are subject to national laws and regulations. In contrast, as it is based on the worldwide Internet, the concept of cloud labor is not limited to specific countries. That implies a need for research on the consolidation of national employment and fiscal laws into a global perspective. Relevant research questions include:

1. Under what conditions can the cloud labor concept be a viable work model that successfully complements traditional work models?
2. How can the social acceptance of the cloud labor concept be fostered? (Kittur et al., 2013)
3. How may the existing national laws and tax models be adapted to the needs of the globalized perspective of the cloud labor concept?

2.4. Application perspective

This section describes the state of the art of cloud labor services from the perspective of the usage scenario. After providing an overview on the existing types of applications in section 2.4.1, section 2.4.2 explains how relevant applications may be identified. Section 2.4.3 then covers concepts and tools for defining task workflows as well as for decomposing tasks into smaller portions of work and recomposing the results. Finally, section 2.4.4 addresses some considerations regarding privacy, copyright and compliance.

2.4.1. Existing applications

This section classifies existing cloud labor service applications in two ways. It examines what applications public cloud labor platforms are actually used for as well as what applications have been investigated in the scientific literature.

Applications on public cloud labor platforms

In order to develop an understanding of the types of applications being implemented on public cloud labor platforms a classification has been created within the scope of this thesis based on a manual analysis of all available tasks on the Amazon Mechanical Turk (MTurk) platform[8] in August 2009. The tasks have been categorized by the action the worker is asked to perform. Table 2.3 presents the resulting classification. The classification consists of five main categories, with each one comprising two to five application types. The category *Create content* comprises any applications in which workers are asked to create textual or multimedia content or provide creative ideas. Category *Revise content* addresses all applications in which workers are asked to classify, refine, summarize or transform content. This includes the identification of pornographic or other undesired content. It also applies to the recognition of objects on images or the transcription of recorded speech. In *Research information* applications, workers are asked to locate and potentially retrieve

[8] See section 2.5.1 for a comprehensive description of the platform.

Table 2.3.: Classification of cloud labor service applications by the activity to be performed by the worker.

Activity	**Type of application**
Create content	Create texts
	Create media
	Creativity or idea generation
Revise content	Check or classify content
	Tag content
	Audio or video transcription
	Summarize content
	Refine content
Research information	Data or information research
	Media research
Rate or sense information	User opinion
	User knowledge
Other activity	Language services or translation
	Programming
	Software or Web application testing
	Traffic augmentation
	Location–specific services
	Other

information or multimedia data from the Internet. In the applications of category *Rate and sense*, users are asked to provide subjective feedback, for example by responding to surveys or polls or by telling about their individual experiences with products or websites. The last category covers all types of applications that do not fall into any of the other categories. At the time of the analysis, the major portion of tasks in that category applied to translation, software and Web application testing. The category also covers location specific applications, in which the worker has to be at a specific location in order to provide the service, for example by taking a picture of a specific event. A remarkable application in this category is *Traffic augmentation*, in which workers are asked to generate traffic or to provide positive feedback in the Web 2.0 environment (e.g. rating a YouTube[9] video, promote people on Facebook[10], increase search rank of websites, etc.). The class *Other activity / other* covers any applications that are not covered by the previous categories or classes and for which no separate class was defined because there had only been a small number of instances.

The analysis indicates, that creating, revising, researching, rating and sensing information are the key categories of applications being performed on cloud labor platforms. This is actually confirmed by the way the platform providers are advertising their services. For example, MTurk focuses on the four categories ”clean your data”, ”categorize items”, ”get feedback” and ”create or moderate content” (Amazon Inc., 2013a).

Interestingly, a number of companies has emerged whose business model would not work without having access to the scalable workforce of cloud labor platforms. One of them is the transcription service CastingWords[11], which will be discussed in detail in section 3.3.5.

[9] `http://www.youtube.com/`, last accessed on July 1, 2013.
[10] `http://www.facebook.com/`, last accessed on July 1, 2013.
[11] `http://castingwords.com/`, last accessed on July 1, 2013.

Applications discussed in the scientific literature

Naturally, cloud labor services are primarily used for tasks that cannot be automated. These tasks represent problems that cannot be satisfactorily solved by *artificial intelligence* (AI) , "the science and engineering of making intelligent machines" McCarthy (2007). The vast majority of applications discussed in the scientific literature can be mapped to such problems. Table 2.4 clusters a selection of research papers according to areas of artificial intelligence.

Table 2.4.: Examples of cloud labor service applications clustered by typical problems of artificial intelligence.

Area	**Application**
Natural language processing	Language translation, evaluation of machine translation quality (Callison-Burch, 2009), Word sense disambiguation (Parent & Eskenazi, 2010), Textual entailment (Negri & Mehdad, 2010), Text creation based on abstract, modification of tense (Little et al., 2009a), Creation of question-answer sentence pairs (Kaisser & Lowe, 2008), Ranking of computer generated questions about provided texts (Heilman & Smith, 2010), Rating Wikipedia articles (Kittur et al., 2008)
Relevance assessment	Relevance of search results (Grady & Lease, 2010), Relevance evaluation (Alonso et al., 2008)
Sentiment analysis	Classifying sentiment in political blog snippets (Hsueh et al., 2009), Obtain polarity scores for customer comments (Mellebeek et al., 2010), Capturing the amount of action indicated by a sentence (Madnani et al., 2010), Web site reviews and marketing surveys (Barr & Cabrera, 2006)
Visual perception	Validate hierarchy of images generated by an algorithm (Deng et al., 2009), Describe a given image in one sentence (Rashtchian et al., 2010), Geometric reasoning experiment: Fold protein structures in a three-dimensional space (Corney et al., 2010), Identify and locate objects in images (Yang et al., 2008), Validate identity of a person (Gentry, 2009), Medical image segmentation (Raykar et al., 2009)
Audio processing	Transcription of voice recordings (Marge et al., 2010; Novotney & Callison-Burch, 2010a; Liem et al., 2011), Elicit narrations of Wikipedia articles as audio recordings (Novotney & Callison-Burch, 2010b), Various applications (Parent & Eskenazi, 2011)
Knowledge provisioning	Ontologies creation and matching (Eckert et al., 2010), Evaluate common sense knowledge from news or Wikipedia (Gordon et al., 2010)

In addition to extending the reach of AI, cloud labor services are also considered an important tool for conducting behavioral research. According to Mason & Suri (2011), surveys performed on task platforms have been observed to deliver results comparable to online surveys. However, they provide access to a more diverse population than traditional subject pools do. Also the wages tend to be lower.

2.4.2. Identification of relevant tasks

Although the classifications of existing usage scenarios provided in the previous section can be used as a starting point when searching for additional use cases, they do not necessarily provide a complete picture. Especially within organizations there might be a large number of promising use cases which are specific to certain industries or methodologies.

From the concept of cloud labor services described in section 2.1.2, a number of basic requirements can be deduced that help identifying relevant applications. First, the task must be issued (but not necessarily performed) through an electronic interface. Second, the entire task invocation cycle is automated. In particular, there is no personal interaction required with the requester. Additional considerations can be made from an economic perspective: Because of the one time effort for setting up the task and developing the worker interfaces, there should be a large number of equivalent tasks. In order to profit the most from the cloud labor service (compared to using dedicated employees), this number should ideally vary over time. And finally, cloud labor services obviously primarily make sense if automating the task is not an option.

Altogether as a rule of thumb, cloud labor should be considered if the task

1. can be mapped to an electronic interface (Web, mobile device, etc.).
2. does not require personal interaction with the requester.
3. is subject to a large (and ideally varying) demand.
4. can either not be automated at all, or not well enough or automating it would be too expensive or too time consuming.

Often, a task can be automated, but the results are actually not good enough. In such situations, cloud labor can be used to accomplish the delta. For example, the recognition of handwritten texts still cannot be perfectly automated[12]. If a high level of quality is needed, such cases that are difficult to decide can be passed to a cloud labor platform in order to increase the overall result quality.

Cloud labor services can also help to reduce the time to market because they can often be set up in a few hours or even faster while automating the task may take much longer. So, even if automation is possible, cloud labor may sometimes be the preferred choice. This especially applies to the ramp-up phase of new services. Once the service is up and running, the focus may be shifted towards automating the service in order to reduce the overall cost. For temporary efforts, it may not be worth considering automation at all, because it may be more expensive than to apply cloud labor services.

2.4.3. Workflows and task granularity

There are many situations in which a sequence of multiple cloud labor tasks needs to be performed or a cloud labor task needs to be combined with automated tasks. Such sequences are called workflows. Workflows may also be used to reduce the task granularity by decomposing complex tasks into smaller, manageable units. Another specific objective of workflows can be to improve the quality of work results by combining activities of multiple workers, e.g. a response generated by a first user is validated by a second one.

There are situations in which it may be useful to pursue both objectives in parallel in order to optimize the overall system. For example, by having one worker compose responses from other workers into a final result, that worker could implicitly validate the responses of the other workers. As the focus of this thesis is quality management, workflows are discussed

[12] Refer to section 8.1.

in two separate chapters: The overall concepts of workflows and their use for reducing the granularity is discussed here, while specific workflows for quality management will be discussed in section 3.3. For example, *TurKontrol* (Dai et al., 2010), which is often mentioned in the context of human computation workflows, is actually rather a quality management concept and is therefore covered later.

Table 2.5 provides a list of the workflow concepts that have been proposed in the context of cloud labor services in the last couple of years.

Table 2.5.: Workflow models and tools for cloud labor services.

Name	**Key features**	**Source**
TurKit	Supports workflows created in JavaScript and provides a tracing and a ”crash-and-rerun” feature.	(Little et al., 2009b)
CrowdLang	Supports predefined workflows as well as dynamic creation of workflows at runtime.	(Minder & Bernstein, 2011)
CrowdWeaver	Allows for visually creating and managing workflows with real time monitoring and notification.	(Kittur et al., 2012)
Crowdforge	Implements MapReduce framework.	(Kittur et al., 2011)
Turkomatic	Allows for recoursively decomposing work into smaller portions.	(Kulkarni et al., 2011)
Clowder	Uses decision theory for deciding on task flow, customizing interfaces and controlling quality.	(Weld et al., 2011)

TurKit (Little et al., 2009b, 2010b) is implemented as a Java program that can execute Javascript files and provides an API to them through which they can communicate with the MTurk platform in order to use ”MTurk workers as subroutines”. Additional functions are available for writing information into a database and for writing trace messages that can be used for tracking the execution flow. A recent crash-and-rerun feature ensures that long running processes can be seamlessly continued after a crash. TurKit can be accessed using a public Web GUI by entering the MTurk credentials.

TurKit has been used as a basis for a series of other research contributions. One example is *Soylent* (Bernstein et al., 2010), ”a word processor with a crowd inside”, that allows for crowdsourcing text editing tasks to MTurk workers out of a Microsoft Word$^{\text{TM}}$document.

Rather than extending a state of the art programming language, *Crowdlang* (Minder & Bernstein, 2011) represents a proprietary executable and model-based programming language and framework that is specifically designed for creating workflows of human tasks. Workflows can either be predefined or created by workers at runtime. The tool comprises ready to use operators for managing task granularity, the actual control flow and the aggregation of responses.

CrowdWeaver (Kittur et al., 2012) does not use a written programming language but is designed as a visual system. Workflows of human tasks and machine tasks are represented as a visual model and are created using a visual interface. At runtime, the workflow graph is being updated in real-time with status information like the number of instances being processed of each task, the quality and the costs. Templates allow for reuse of workflows. A notification feature keeps the user informed about important events. The tool is implemented by a JQuery-based Web visual and a Ruby controller and internally uses a MySQL database. In contrast to the tools discussed previously, CrowdWeaver is not based on MTurk but on the CrowdFlower[13] platform. That way, it has access to a wider

[13] The CrowdFlower platform will be described in section 2.5.1.

range of Cloud labor platforms, including MTurk. CrowdFlower functionality is used to create and modify the underlying human tasks.

CrowdForge is a concept and prototypical system for managing task granularity. The basic idea is to crowdsource not only the actual task execution but also the breakdown of complex tasks into manageable units and the recombination of the individual results. Inspired by Google's MapReduce algorithm, the entire process consists of three steps: A "partitioning task", a "map task" and a "reduce task" are each being defined as a separate human sub-task. A first set of workers recursively breaks down the work into smaller units, which are then executed by other workers. Again other users are finally recombining the work results on several stages until the overall result has been generated. In addition to task de- and result re-composition, CrowdForge (Kittur et al., 2011) also supports a variety of quality management mechanisms. The tool is implemented as a web-based prototype.

Turkomatic (Kulkarni et al., 2011, 2012) suggests a recursive approach similar to the one used by CrowdForge. A collaborative approach is proposed for managing the quality of the decomposition and recomposition tasks.

Weld et al. (2011) argue that "artificial intelligence methods can greatly simplify the process of creating and managing complex crowdsourced workflows". Their *Clowder* framework represents a vision of an all purpose cloud labor system that uses decision theory for deciding on alternative workflows, for personalizing the appearance of interfaces and for controlling the overall workflow. They expect that "optimized workflows are significantly more economical (and return higher quality output) than those generated by humans". One of the features they suggest is the personalization of interfaces based on implicit "A-B" testing performed by workers. They also recommend the use of a hierarchical task-network (HTN) for modelling task decomposition.

2.4.4. Privacy, copyright and compliance

The cloud labor concept is based on the exchange of information. Information is being passed to the workers through the cloud labor services platform as input for the task. Based on this information, the worker processes the task, potentially by utilizing external information from the Web or, especially in case of surveys, by using information about personal preferences, desires etc. While working on the task, the worker may generate new intellectual property. For example, a text authored by the worker will be subject to copyright. There may be four major types of critical information involved in the cloud labor service scenario, which are illustrated by figure 2.4:

1. Sensitive information of the requester's organization. This could be business secrets or any information that is subject to data protection acts or other rules and regulations. An example is sensitive information about customers, employees or business partners.
2. Personal sensitive information of the workers, e.g. details about their financial situation or their personal desires.
3. Intellectual property generated by the workers either during task execution or earlier.
4. Copyrighted information used by the workers during task execution, for example a text or picture being retrieved from a website.

The potential exposure of sensitive information or use of protected intellectual property causes a series of risks that need to be addressed accordingly. The most obvious risk is

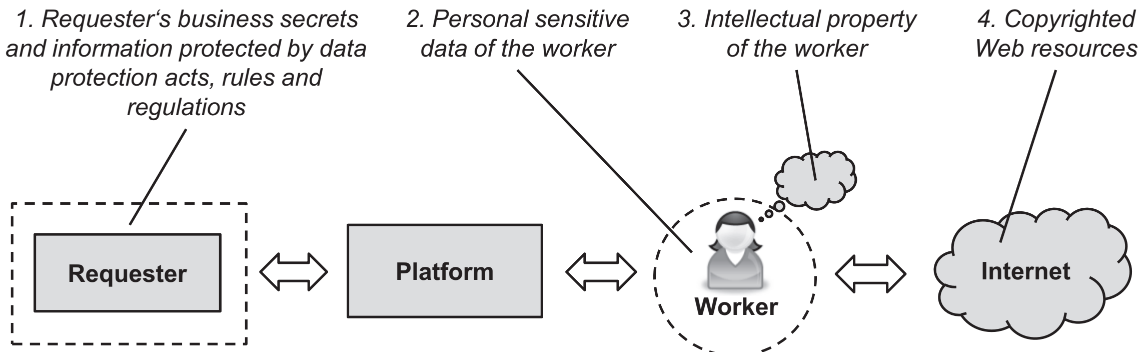

1. Requester's business secrets and information protected by data protection acts, rules and regulations
2. Personal sensitive data of the worker
3. Intellectual property of the worker
4. Copyrighted Web resources
Requester
Platform
Worker
Internet

Figure 2.4.: Critical information potentially being involved in the cloud labor service scenario.

probably the disclosure of sensitive information by the requester. A common mitigation strategy is the anonymization of the data by removing or replacing all irrelevant information like names and phone numbers, or by splitting tasks into small units. For example, the transcription service CastingWords[14] breaks speech recordings down into small pieces before they are transcribed into text. Afterwards, the pieces are recomposed into a complete transcript. As most workers see just small portions of the transcript, the risk of privacy issues is reduced[15]. However, anonymization may not fully avoid the risk of disclosing sensitive information. As Felstiner (2011) points out, a "crowd worker may still be able to glean knowledge of a valuable piece of intellectual property by completing even a small task". This risk increases with the number of closely related tasks a worker performs.

If the information cannot be anonymized, a private or hybrid people cloud can be used. The most conservative approach would be to use a workforce from the requester's organization. For less critical privacy issues it might be sufficient to oblige service workers to maintain confidentiality about the processed information by adding an appropriate clause to the terms and conditions of the task or platform.

From the worker perspective, there is a risk of potential misuse of personal data. Even though the workers' identity is usually not disclosed to the requester, a unique ID is being assigned to each worker which is attached to any responses returned by that worker. Over time, the requester may gather a large amount of information about the worker based on that ID. As long as the worker does not explicitly trust the requester, a basic mitigation strategy is not to disclose any personal information. Platform providers should not disclose the identity of the workers.

If the task involves the creation of intellectual property by the worker, the requester and / or the platform provider should ask the workers to grant their IP rights to the requester. In this context, an important risk is the misuse of copyrighted information, e.g. due to plagiarism. This applies specifically to content generation applications. Instead of creating new content, workers may search for appropriate content in the Web and represent it as their own original work. If the requester then uses it publicly, he might get sued by the actual originator. Even though penalties could be enforced against the worker for such an offense, this would likely not remedy the damage, at least not in public people cloud scenarios in which there is only a lose contract with the worker. Therefore, the requester

[14] CastingWords will be discussed in detail in section 3.3.5.
[15] `http://castingwords.com/support/transcription-faq.html`, last accessed on March 11, 2012.

should check for plagiarism by using services like Plagiarism-Detect[16] or Copyscape[17] or by asking other workers to manually perform a check.

In information research applications, licensing issues are another possible threat. For example in an image research application a requester might ask the workers to find pictures of products that he wants to use for an online product catalog. The requester needs to ensure that the licensing rules of those images allow for commercial use. Therefore, he should always ask the worker to return the link for the image instead of the image itself. This way, the licensing rules can be validated, for example by asking other workers to do so.

2.5. Platform perspective

This section describes the state of the art of cloud labor services from the perspective of the coordination platform. After providing an overview of the existing commercial cloud labor platforms in section 2.5.1, section 2.5.2 introduces considerations about their service models and technical infrastructure. Section 2.5.3 then addresses the matching of workers to tasks and tasks to workers while section 2.5.4 finally refers to quality management for cloud labor services.

2.5.1. Existing platforms

A large number of cloud labor platforms have emerged in the last few years. The industry website Crowdsourcing.org (2012b) lists 164 entries in the category of cloud labor[18]. Compared to that, Frei (2009) had listed only 46 sites in 2009. As most of the platforms do not disclose any information about their task throughput, it is difficult to estimate the actual relevance and ranking of the sites. In order to identify the key players, table 2.6 contains those of them that have been sponsors or exhibitors on the 2010 CrowdConf (Crowdflower, 2010), which can be considered the primary commercial crowdsourcing event of the year. As the only exception, the table also covers the MTurk platform which is used as the workforce provider for several of the other platforms. The table only captures platforms for programmatic cloud labor and does not cover platforms for manually managed cloud labor like ODesk. The last two columns indicate the founding year and the number of workers available to the platform followed by the type of workforce. "Own" means that the workforce is managed by the platform itself, "MTurk" means that the workforce of MTurk is used. "Several" indicates that the platform relies on the workforces of several other platforms.

One of the most popular cloud labor platforms is MTurk which has been started in late 2005. On MTurk, service requesters can publish open calls for *human intelligence tasks* (HITs). Any Internet user that meets certain skill criteria may act as a service worker and work on tasks to earn some money. Amazon keeps a fee of 10 percent of the task price with a minimum of \$0.005 per task[19]. MTurk provides a basic Web interface for interactively defining tasks and designing the corresponding user interfaces. In addition, a command line API as well as SOAP and REST based APIs support several programming languages

[16] `http://plagiarism-detect.com/`, last accessed on July 1, 2013.
[17] `http://www.copyscape.com/`, last accessed on July 1, 2013.
[18] The directory does not just comprise crowd labor platforms but also other sites that build there business model on the concept of cloud labor.
[19] See FAQ section on the help page of (Amazon Inc., 2013a).

Table 2.6.: Selection of commercial cloud labor service platforms.

Platform	**URL**	**Focus**	**Country**	**Year**	**Workers**
Amazon Mechanical Turk	mturk.com	All purpose platform with self service only and limited quality contol; used as a workforce provider by many other platforms	USA	2005	500k (own)
Clickworker	clickworker.com	Processing of unstructured data, such as text, photographs, and videos	Germany	2005	300k (own)
CrowdFlower	crowdflower.com	All purpose platform with specific focus on quality management	USA	2007	3M (several)
Microtask	microtask.com	Document processing and data entry	Finland	2009	unknown
Microworkers	microworkers.com	Traffic augmentation	USA	2009	unknown
CloudFactory	cloudfactory.com	All purpose platform with focus on ongoing processes as well as one-time tasks	Nepal	2011	500k (Mturk)
Crowdsource	crowdsource.com	All purpose platform	USA	2010	500k (Mturk)
Serv.io	serv.io	Content creation, content management, translation services	USA	2009	150k (own)
UTest	utest.com	Application testing (Web, mobile and desktop)	USA	2007	70k (own)

including Java, Ruby, Perl and .NET[20]. While workers from the US and from India can transfer their earnings to a bank account, workers from all other countries can claim their earnings only in form of Amazon gift certificates (Amazon Inc., 2013b).

According to Ipeirotis (2010a), MTurk is a ”heavy-tailed market” with regard to the requester activity. A minority of one percent of the requesters account for half of the overall dollar amount being paid to the workers. Furthermore, there is a focus on small tasks. For 90 percent of the HITs a reward of $0.10 or less is being paid.

Another early player is the German platform *Clickworker* who first went live in 2006 under the name *Humangrid*, which is still the name of the company that operates the platform. Clickworker has a workforce of about 300.000 own workers and mainly focuses on processing unstructured information, such as text, photographs, and videos (Humangrid GmbH, 2013).

The *Crowdflower* platform founded in 2007 positions itself as the market leader among the platforms for cloud labor services. It is a Meta platform that leverages the workforce of several other platforms and external workforce providers including MTurk. According to an e-mail of the founder Lukas Biewald, they process about one million tasks on an average day, corresponding to an effort of 4 person years (Biewald, 2012).

Microtask from Finland primarily concentrates on document processing and data entry

[20] See developer section of the MTurk requester website (Amazon Inc., 2013a).

with a specific focus on digitizing handwritten forms. In cooperation with the National Library of Finland, volunteer workers are leveraged to correct OCR errors in a historical newspaper archive (Microtask Oy, n.d.).

The *Microworkers* platform launched in mid 2009 mainly focuses on traffic augmentation applications like search engine optimization. Compared to MTurk, two additional important differences concern the way payments are being made to the workers as well as the structure in which the tasks are being organized. While MTurk requires a US bank account for monetary transactions, Microworkers supports online payment services like PayPal[21], which are better suited for international workers. Rather than putting all HITs into a single list, Microworkers provides predefined job categories with individual minimum payments starting from \$0.10. (Hirth et al., 2011)

CloudFactory was founded in 2011 in Nepal in order to develop job opportunities for the developing countries. The platform *Crowdsource* was acquired by ScalableWorkforce LLC in 2011 and merged with their own platform into crowdsource.com (CrowdSource, n.d.). *Serv.io* claims to be the market leader for e-commerce content services[22]. Through their ClowdCrowd platform they have access to over 250k workers[23].

UTest founded in 2007 is not a typical task platform but an application testing service for Web, mobile and desktop applications that uses crowdsourcing in order to distribute testing work to a pool of some 70k testing professionals around the world[24]. Because of the formalization of those tasks and the limited need for personal interaction with the requester, these tasks are also considered to match the definition of cloud labor services.

2.5.2. Technical infrastructure

Because of the analogies between cloud computing and cloud labor it can be assumed that similar service models can be applied to both concepts. This section aims to provide a sketch of a cloud labor stack by deriving a set of service models from those of cloud computing, which are defined by Mell & Grance (2011) as:

- *Infrastructure as a Service* (IaaS) provides the actual IT infrastructure including computing power, storage and networks on which the user can deploy operating systems or other arbitrary software. The user only manages the operating system or other software but has no control over the underlying infrastructure.
- *Platform as a Service* (PaaS) allows for deploying own applications or third party applications on top of the cloud infrastructure by leveraging programming models supported by the provider. The user only controls the "application-hosting environment" but has no control over the underlying operating system or hardware.
- *Software as a Service* (SaaS) provides remote access to specific applications that are running on a cloud infrastructure. The user only controls "user-specific application configuration settings" but has no control over the underlying hosting environment, operating system or hardware.

Inspired by the definitions of Mell & Grance (2011), a similar stack could be described for cloud labor which is illustrated by figure 2.5:

[21] `https://www.paypal.com/`, last accessed on July 1, 2013.
[22] `http://www.serv.io/servio-announces-record-quarter-97-quarter-over-quarter-growth-average/`, last accessed on April 2, 2013.
[23] `http://www.cloudcrowd.com/company/about`, last accessed on April 2, 2013.
[24] `http://www.utest.com/about`, last accessed on April 2, 2013.

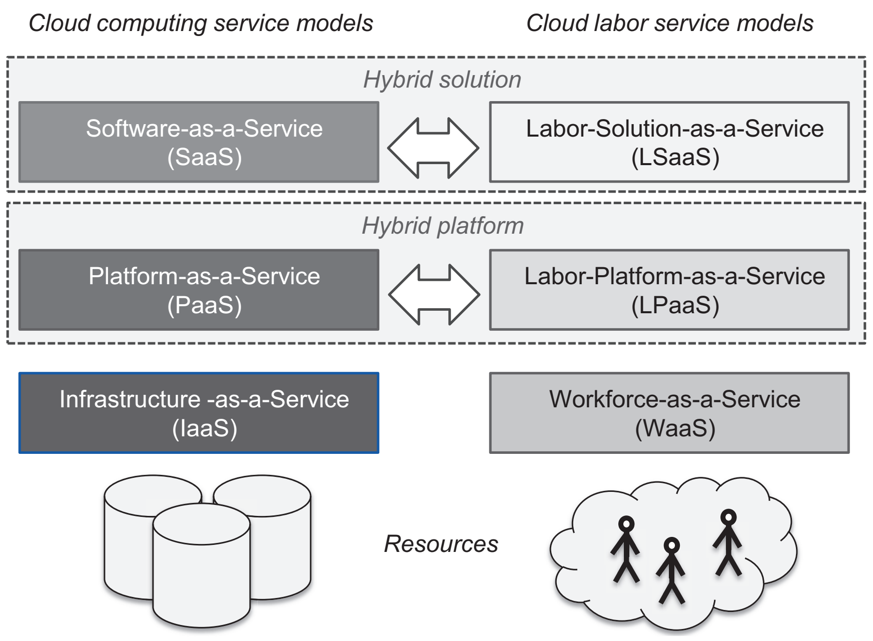

Cloud computing service models
Cloud labor service models
Hybrid solution
Software-as-a-Service (SaaS)
Labor-Solution-as-a-Service (LSaaS)
Hybrid platform
Platform-as-a-Service (PaaS)
Labor-Platform-as-a-Service (LPaaS)
Infrastructure -as-a-Service (IaaS)
Workforce-as-a-Service (WaaS)
Resources

Figure 2.5.: Concept of a cloud labor service (cloud labor) stack in comparison to the cloud computing stack.

- *Workforce as a Service* (WaaS) delivers the workforce of a crowd of workers to a requester as a scalable resource. The requester can define arbitrary tasks which are completed by the workers remotely, typically through a Web application or mobile application, and for which responses are delivered by the workers back to the requester. The requester does not manage or control the availability of the individual workers but can decide, which workers are allowed to work on specific types of tasks.
- *Labor Platform as a Service* (LPaaS) provides the ability for requesters to deploy cloud labor solutions that have been created by using workflow languages and task definition capabilities provided by the platform. The consumer does not manage the assignment of workers tasks but has control over the quality of work results and the skill profiles of the workers by utilizing facilities provided by the platform.
- Labor Solution as a Service (LSaaS) provides access to the cloud labor solutions running on a labor platform. The cloud labor solutions are accessible from various client devices such as a Web browser or API. The requester does neither manage or control availability, assignment or the skill of the workers, nor defines the quality of the work results, with the possible exception of solution specific configuration settings.

Of course, there are many applications that require both cloud labor and computing resources. Therefore, there should not be a sharp borderline between SaaS and LSaaS. Instead, LSaaS should be merged with SaaS into a hybrid layer in order to be able to effectively combine the capabilities of both worlds. The resulting hybrid solutions may again be running on a hybrid platform which itself is a merger of PaaS and LPaaS. This layer would have access to both IT infrastructure (IaaS) and human workforce (WaaS).

The above considerations have been developed based on a discussion with Stephen Dill from the IBM Almaden Research Center in California, USA in March 2012. A similar stack was proposed independently by Panos Ipeirotis in July 2012 (Ipeirotis, 2012) which mainly differs in two aspects: First, from a terminology perspective, the term *Labor Infrastructure as a service* (LIaaS) is used instead of WaaS. In this thesis, WaaS is preferred in order to emphasize the fact that the actual service being provided to the requester is the workforce of the crowd. Second, LSaaS is defined by Labor Applications / Software as a service and represents basically what is described as hybrid solution above. In order to clearly distinguish from and illustrate the symmetry with the cloud labor stack, LSaaS is defined independently of the cloud labor stack here.

The existing commercial cloud labor services platforms follow different service models. MTurk can mainly be seen on the level of WaaS, but also cooperates with partners in order to deliver hybrid solutions, e.g. for media tracking. MTurk actually names such solutions ”Apps”.[25] Crowdflower could be seen as LPaas because it leverages a multitude of WaaS providers including MTurk. Crowdflower also offers a series of LSaaS or hybrid solutions, e.g. for content moderation, lead data enhancement and sentiment analysis.[26] However, as Turian (2012) points out, Crowdflower does not provide native support for workflows. He sees MobileWorks close to the idea of LPaaS but mentions that they offer only immature self-service capabilities. On the LSaaS level, he sees CastingWords, UTest, Microtask and others, but he underlines that due to the lack of a robust LPaaS provider, all labor solutions ”must be vertically integrated with the platform layer or implement their own application-specific platform”.

These observations indicate that there is indeed an overlap between the concepts of cloud computing and cloud labor. However, the option for deploying own solutions on cloud labor platforms in the sense of PaaS or hybrid platforms is still limited to explicit cooperations with the platform providers. The functionality is not yet available as a self service according to the original idea of cloud computing.

From an architectural and implementation perspective, the considerations about BPM and SOA mentioned in section 2.2.6 are highly relevant for cloud labor services as they provide fundamental standards and tools for orchestrating human tasks.

2.5.3. Worker-to-task matching and allocation

In the context of cloud labor platforms, *matching* is the process to identify suitable tasks to be performed by a worker or to identify suitable workers for a task. *Allocation* is the actual assignment of a task to a worker. The following paragraphs describe how the two concepts are being addressed.

Allocation

According to the definition of crowdsourcing, the allocation is always performed as an open call, i.e. the worker browses tasks and decides what tasks to work on. In practice, this open call is often not fully open, but the pool of workers who are eligible to work on a task is restricted. This can be due to language, location or skill requirements or because a so called *certification test* needs to be successfully completed before being allowed to work on a task. It can be argued, ”how open” the call needs to be in order to satisfy the definition of crowdsourcing. In any case, a directive assignment of a task to a worker would not be considered crowdsourcing.

[25] See apps and case studies on (Amazon Inc., 2013a).

[26] See solution overview on `http://crowdflower.com/products`, last accessed on February 6, 2013.

The actual assignment of a task could also be the result of an auction. Satzger et al. (2011a) describe a concept in which a closed-bid auction mechanism assigns tasks to one of multiple workers who have bid for it. Even though the actual assignment is being performed by the platform, there is still an open call which in this case is a call for bids.

Apart from the possible use of auction mechanisms, the open call assumption leads to the fact that the allocation of workers is rather straightforward. The actual challenge is the matching step, which comes down to presenting the right options to the worker to choose from.

Matching

There are three factors that constitute the list of tasks that is presented to the worker:

1. A search term and a sort order specified by the worker.
2. A qualification requirement or other restriction that prevents the worker from working on the task.
3. A recommender system that proposes suitable tasks to workers.

On current task platforms like MTurk, only options 1 and 2 are implemented. Workers typically browse for the most recently posted tasks or the ones that have the largest number of available instances and limit their scope to the first pages of the task list (Chilton et al., 2010). For the requesters, this behavior limits the chance for getting access to workers with a specific expertise. It also reduces the average result quality as many workers are performing tasks for which they do not have the skills (Ambati et al., 2011).

For the workers, there can be an advantage to focus on tasks with qualification requirements as this will likely reduce the competition of workers. DiPalantino et al. (2011) have observed that the revenue tends to be smaller if there are many competitors. Better matching mechanisms also have the potential to reduce the costs and the resolution time. Chilton et al. (2010) have observed that tasks which are easy to find for a worker are "completed 30 times faster and for less money" than others. Therefore, recommending suitable tasks to workers (option 3 mentioned above) can be assumed to be an important ingredient for cloud labor platforms. However, recommender systems have only rarely been described in the context of crowdsourcing and specifically cloud labor.

For Wikipedia, Cosley et al. (2007) have developed a *Suggestbot* that proposes articles to be revised by editors. In a preprocessing step, the software creates a local mirror of Wikipedia that contains the texts of all articles and identifies all editing needs based on tags provided within the articles. At the same time, it models the interests of the Wikipedia editors by identifying all editing activities they have previously performed and all the links they have created between articles. By analyzing the text similarity to other articles that need work, recommendations are made to the editors.

For cloud labor platforms, only a small number of papers have proposed initial ideas about the use of recommender systems: Ambati et al. (2011) have sketched a recommendation engine for suggesting tasks to users based on implicit modeling of skills and interests. The system creates a preference model of the worker using a bag-of-words approach. The model is based on three types of information:

- A profile created by the worker covering information about his location, education, skills and experience,

- explicit feedback about the worker's rating of the tasks and the payment, and
- implicit feedback using search terms, tasks being selected, task description, reward, number of instances and the requester's feedback in form of rejection, bonuses or comments.

Yuen et al. (2011) have proposed a system that assumes a platform with well defined task categories. By keeping performance data per category like selection preference, task acceptance rate and reward and time alloted, the system aims to rank the available tasks according to the order of best matching.

Matching of workers may also be influenced by their locations. For example, Reddy et al. (2010) have described a "recruitment framework for participatory sensing data collection" that utilizes the ability of modern mobile phones to identify their location and to be able to perform in-situ data collection by taking pictures, audio or video recordings.

2.5.4. Quality management

As mentioned before, quality management has received most attention of all research foci in the context of cloud labor services (Kittur et al., 2013) and is regarded as the by far most pressing challenge in this space (Crowdsourcing LLC, 2012; Turian, 2012). Because of its importance and because it represents the focus of this thesis, a separate chapter is devoted to this topic. Chapter 3 will solely concentrate on quality management considerations for cloud labor services.

2.6. Workforce perspective

The workforce perspective covers the aspects and concepts that are mainly influenced by the workers. After illustrating the demographics of the workforce in section 2.6.1, section 2.6.2 describes the primary motivators for people to perform work on cloud labor platforms. Sections 2.6.3 and 2.6.4 then address the importance of worker education and feedback mechanisms as well as considerations about the task design. Finally, section 2.6.5 discusses to what extent cloud labor can be considered a desirable work model.

2.6.1. Worker demographics

By far most studies on worker demographics have been performed on the MTurk platform, which is therefore used as an example here. Over the years, the demographics have shifted towards a more international population on that platform. In late 2008, the population had still been US centric. Compared to 83% US workers there had been only a minority of 5% of workers from India and 12% from other countries (Ipeirotis, 2008). At the end of 2009, the portion of Indian workers had already increased to 36%, while there had been 56% workers from the US and 8% from other countries; at the same time, "the number of lower-income workers has increased, along with the number of young workers and male workers" (Ross et al., 2010).

According to Ipeirotis (2010b), the demographics differ considerably between the US and India. In the US, the average age of the workers is some 34 years compared to 27.5 years in India. While 35% of the US workers have at least a bachelor degree, that is the case for 52% of the Indian workers. In the US, 75% of the workers have an annual household income of at least $25,000 and only 13% have less than $10,000, whereas the picture is almost the opposite in India: Only 16% of the workers have an annual household income of more than $25,000 and more than 55% declared to have less than $10,000. In both the US and India, the majority of workers are singles and do not have children. Most of the workers spend only a day or less per week on cloud labor and earn often less than $20 during that time. A portion of 12% of the US and 24% of the Indian workers state that Mturk is their primary source of income, whereas 59% of the US workers and 70% of the Indian workers consider MTurk as a "fruitful way to spend free time and get some cash (e.g., instead of watching TV)" (Ipeirotis, 2010b).

One reason why most of the workers are from the US and from India may be that on MTurk, cash payment is only supported in US dollars and Indian Rupees (Mason & Suri, 2011). And indeed, on the Microworkers platform, which allows for international cash payments through payment services like Paypal, there are much more workers from other countries. According to an analysis performed by Hirth et al. (2011) based on anonymized data received from the platform provider, most workers are from Indonesia (18%), Bangladesh (17%), India (14%), United States (11%).

2.6.2. Motivation and incentives

The factors motivating people to perform work on cloud labor platforms can be divided into two groups, intrinsic and extrinsic motivation (Kaufmann et al., 2011; Rogstadius et al., 2011).[27] Kaufmann et al. (2011) state that "intrinsic motivation exists if an individual is activated because of its seeking for the fulfillment generated by the activity (e.g. acting just for fun)" while "in the case of extrinsic motivation the activity is just an instrument for achieving a certain desired outcome (e.g. acting for money or to avoid sanctions)".

According to Kaufmann et al.'s (2011) model for worker motivation in crowdsourcing presented in table 2.7, there is a set of motivating factors that can be classified to be extrinsic or intrinsic. Each of the categories may be measured by a number of motivational constructs. The score value provided by the last column was determined by a survey performed on the MTurk platform. A higher score means that the construct had been considered more relevant by the workers. The minimum possible value is -0.78, the maximum is 3.99.

Table 2.7.: Model for worker motivation in crowdsourcing with empiric scores for motivational constructs; derived from (Kaufmann et al., 2011).

Type	**Category**	**Construct**	**Score**
Intrinsic	Enjoyment based	Skill variety	2.4
		Task identity	2.3
		Task autonomy	2.4
		Direct feedback from the job	2.0
		Pastime	2.1
	Community based	Community identification	2.0
		Social contact	1.3
Extrinsic	Immediate payoffs	Payment	3.0
	Delayed payoffs	Signaling	1.9
		Human capital advancement	2.2
	Social motivation	Action significance by external values	1.7
		Action significance by external obligations & norms	1.0
		Indirect feedback from the job	1.7

With a score of 3.0, the extrinsic construct *payment* sticks out from all other constructs. That reveals that payment is the primary motivation for people to do work on MTurk.

[27] This distinction was originally proposed in the self-determination theory by Deci & Ryan (1985).

But also task related factors seem to be important. Those factors include the variety of skills needed for performing the task, the autonomy to decide how to perform the work, the direct feedback that workers may get by validating the quality of their work themselves[28], and also the opportunity to train their skills or gather new skills by working on a task (*human capital advancement*).

Interestingly, even killing time (*pastime*) can be a reasonable motivation for using the platform, a phenomenon already identified by Brabham (2008) and by Ipeirotis (2010b). In a survey conducted by Ipeirotis (2010b), 32% of the US workers and 5% of the Indian workers have reported that they participate on MTurk to kill time. However, (Kaufmann et al., 2011) have determined that those workers are not using the MTurk platform frequently.

The authors also point out the ”high potential for community induced intrinsic motivation” and in fact, even though the MTurk platform does not provide any capabilities for community building, a number of external communities like Turkernation[29] or the MTurk forum[30] and tools like Turkopticon (Silberman et al., 2010) have emerged that provide a forum for workers to discuss experiences and comment on requesters.

Social motivation like altruism (*action significance by external values*) or seeking for commendation (*indirect feedback from the job*) seem to be less relevant for the workers. This may be a specific characteristic of tasks platforms on which the workers are usually acting anonymously. External obligations usually do not exist on Mturk because the workers are freely choosing the tasks they are working on rather than being forced by an employer to work on specific tasks. That can be assumed to be different in an enterprise crowdsourcing scenario.

2.6.3. Education and feedback

Education and feedback obviously play an important role in the cloud labor service scenario because the availability of adequately skilled workers is the primary requirement for generating high quality results. For certain tasks types, a specific training may be required in order to prepare the worker for successfully completing it. Such a training is usually integrated into a qualification test that the worker has to complete before being allowed to work on the tasks. On most platforms workers can contact the requester via e-mail if they have questions regarding the task. Ambiguous task instructions may also be discussed in worker communities like TurkNation. However, in order to prevent fraud, only the instructions themselves and not the responses to individual tasks may be discussed.[31]

Providing regular feedback to the workers can be seen as another important ingredient for continuously improving their skills. Hattie & Timperley (2007) differentiate between four general types of feedback, two of which can be assumed to be relevant for cloud labor services: *Feedback about the task* and *feedback about the processing of the task.*

The most common form of feedback is feedback about the task which ”includes feedback about how well a task is being accomplished or performed, such as distinguishing correct from incorrect answers” (Hattie & Timperley, 2007). A major drawback is that feedback about the task often cannot be easily generalized to other tasks and, therefore, does not

[28] For example, in a programming task, direct feedback could be received if the workers test the code that they have developed.
[29] `http://turkernation.com`, last accessed on July 1, 2013.
[30] `http://mturkforum.com`, last accessed on July 1, 2013.
[31] `http://turkernation.com/faq.php?faq=vb3_board_faq`, last accessed on February 17, 2013.

encourage the receiver to develop strategies to attain a goal. Feedback about the processing of the task addresses this limitation by enabling the receiver to detect potential errors independently and to adjust the strategy accordingly, possibly by seeking help.

Platforms like MTurk support feedback about the task by allowing the requester to accept or reject responses along with a comment about the rationale for the decision[32]. Feedback about the processing of the task is partially supported by some quality management mechanisms. For example, Le et al. (2010) are using training tasks with predefined explanations of potential errors. This will be explained in more detail in section 3.3.2.

Not just the type of the feedback matters but also its timing. In a study with high school students, immediate feedback has been observed to be more effective for simple tasks while delayed feedback is still effective for complex tasks. This suggests that difficult tasks require more feedback regarding the processing of the task, which can be better provided in form of delayed feedback. (Clariana et al., 2000)

2.6.4. Task design

This section summarizes a number of observations that may be used as a guideline when designing cloud labor services and when deciding on the type and amount of the payment.

In an empirical study on the MTurk platform, Schulze et al. (2011) found out that in general, workers prefer clear instructions, genuine requests and a task design which does not require them to leave the Mturk site for completing the task: Everything they need should be integrated into the task's user interface. One reason the authors mention for that is that many workers have suffered from malicious websites.

The study further suggests that workers can be categorized into three groups which are outlined in the following:

- *Quick profit jobbers* prefer rather simple tasks with a short description that can be completed in a short time and result in a relatively high reward per hour.
- *Informed workers* prefer well described tasks that have been submitted from well-respected requesters who are willing to respond to questions if needed. The tasks should be underpinned with comprehensive examples for correct and incorrect responses so the workers will understand under what conditions their work will be rejected.
- *Challenge seekers* prefer challenging tasks that sound interesting and enjoyable and for which background information is provided. There should be multiple tasks of the same type available and bonuses should be payed for good performance.

Interestingly, the level of education does not seem to have an impact on the worker's preferences. Indian workers seem to be keen on bonuses being paid for good work while US workers and also full-time workers from any region appreciate to work for well-respected requesters (Schulze et al., 2011).

From these observations it can be deduced that the granularity of simple tasks should rather be low, which, as will be discussed in more detail in section 3.1.2, can also have a positive impact on the result quality. For more complex tasks, background information and comprehensive examples should be provided, complemented by the ability to contact

[32] `http://aws.amazon.com/documentation/mturk/`, last accessed on February 17, 2013.

the requester. A good reputation of the requester matters for complex tasks and will for several reasons also have a positive impact on the execution performance: It attracts full-time workers and workers from different regions and, as further discussed in section 2.6.5 also reduces the risk from being put on a black list by worker communities which would drastically slow down task execution. An important factor of the requester's reputation is the payment in terms of the average hourly rate determined by the workers. As described in section 3.1.2, the payment also directly affects the overall execution performance but not the accuracy of the result. However, no payment at all may lead to a higher accuracy than a low payment.

In order to help requesters determining the actual payment, Horton & Chilton (2010) have proposed a model that predicts how workers supply work to a cloud labor platform depending on the available tasks and their expected hourly wage. The model is based on the assumption that workers would not start working on any task as long as the expected wage is lower than a certain minimum wage which the authors define as *reservation wage*. Although their tests have confirmed their model only to a certain extent, the authors are convinced that it still provides a useful approximation. A possible explanation they offer for the mismatch is that a significant number of workers may act as *target earners* who try to work towards a certain self-imposed target wage rather than responding to the actual market, i.e. "when wages are high, a target earner works less".

2.6.5. Work model

From the perspective of the workers, the concept of cloud labor provides exceptional opportunities but also severe risks. The concept offers an unprecedented freedom to decide when, where, how much to work and what to work on. There is no need of commuting to an office during rush hour but at the same time, working remotely does not necessarily mean that people have to work in an isolated manner from home. Along with the success of cloud labor platforms, a new type of flexible office space is emerging that can be booked on an hourly basis. Liquidspace[33] connects "nomadic workers and property owners who have underutilized space". Office space may be booked and paid in seconds using a smartphone application. The booking confirmation displayed on the screen represents the admission ticket for the office. Interestingly, the service is not only used by crowd workers but also by large companies who want to offer flexible offices to their remote workers. Available space includes underutilized desks of large corporations, conference rooms in hotels and even free space in public libraries. (Kolodny, 2012)

On the downside, employees are afraid that their jobs may be turned into freelance cloud labor engagements with no job security or benefits plan. Felstiner (2011) mentions that "many creative professionals and providers of skilled services (such as software development) fear that their industries will go the way of stock photography". In addition, he sees a risk of continuously low wages because "treating individual workers as sources of cheap labor has hardly proven to be a recipe for failure in the past. If some are willing to work for substandard wages and benefits without legal protection, by necessity or choice, there is no reason to believe that their willingness will evaporate in cyberspace". Felstiner points out that existing protective statutes[34] do not adequately cover the concept of cloud labor because "in the past, a worker could not physically perform a unit of piece-rate labor in

[33] `http://liquidspace.com`, last accessed on February 16, 2013.

[34] In his paper, he specifically focuses on the US and discusses to what extent the *Fair Labor Standards Act* (FLSA) and the *National Labor Relations Act* (NLRA) are applicable to crowdsourcing.

under a second. Parties did not make employment contracts from the other side of the planet with the click of a mouse. Employees were unlikely to have twenty-five separate employers in the course of a single workday.” Therefore, he believes that legal practice ”must revise the definition of ’employer’ and ’employee’ to recognize the economic realities of online and virtual work”.

IG Metall, which is the largest[35] labor union in Germany, is concerned about how to represent the crowd workers in a global cloud labor scenario given that they are today organized by industry and location (IG Metall, 2013). Specific challenges the labor union foresees include the definition of agreed wages along with standardized labor conditions and also the question, who would be the bargaining partners. Would it be the platform providers or rather large enterprise requester companies? IG Metall also sees a chance that self-organization of the communities may strengthen the position of the crowd workers with regard to the requesters.

And in fact, as mentioned in section 2.6.2 with Turknation and Turkopticon, first shapes of such communities are already emerging. Therefore, Felstiner (2011) summarized that ”in the end, crowdsourcing relies on the crowd for its very existence. Legal intervention can only buttress and protect the organized efforts of crowd workers; it cannot replace those efforts”.

Chapter 2 has shown that cloud labor services are still an evolving field of research. It poses a variety of challenges that require additional research in a several disciplines. As just discussed, this concerns a critical reflection of the effect cloud labor services will have on the way we will perform work in the future. However, this also concerns their technical and conceptual feasibility. The remainder of this thesis contributes to the latter by focusing on quality management for cloud labor services.

[35] `http://www.dgb.de/uber-uns/dgb-heute/mitgliederzahlen/2010`, last accessed on April 3, 2013.

3. Quality management for cloud labor services

According to a 2012 analyst report on cloud labor platforms, the quality of the work results represents a ”key disruptive opportunity that crowd labor platforms can exploit and improve” (Turian, 2012). This chapter provides a comprehensive overview of the state-of-the-art of quality management for cloud labor services and provides a gap analysis with regard to the objectives of the thesis.

After identifying relevant quality metrics and influencing factors for cloud labor services in section 3.1, section 3.2 provides an overview on the core concepts for quality management of cloud labor services. Section 3.3 then describes the most relevant approaches in detail. Finally, the approaches are compared in section 3.4 and gaps are identified.

3.1. Quality of cloud labor services

Section 3.1.1 defines the term quality as used in the context of this thesis. Section 3.1.2 then identifies relevant quality metrics for cloud labor services. Afterwards, the factors influencing the quality of cloud labor services are examined in section 3.1.3.

3.1.1. Quality

Quality is not an inherent attribute of a product or a service but can only be determined by considering the expectations of the requester. Montgomery (2008, p. 5) expresses that relation by stating: ”The traditional definition of quality is based on the viewpoint that products and services must meet the requirements of those who use them”. De Feo & Juran (2010, p. 5) reduce it to the simple formula: ”Quality is fitness for purpose”. In this thesis, the term quality is defined according to the ISO standard 9000:2005 as ”the degree to which a set of inherent characteristics fulfills requirements” (ISO - The International Organization for Standardization, 2005). According to this definition, quality is optimal if the requirements are perfectly met. This point of view anticipates that if quality is not optimal, there must be a gap between the actual characteristics of the service and the expectations of the user. According to the GAP model developed by Parasuraman et al. (1985) ”the quality that a customer perceives in a service is a function of the magnitude and direction of the gap between expected service and perceived service”.

Obviously, setting the right expectations is one of the key ingredients for providing good quality of service. For IT services, this is typically being done by using so called *service level agreements* (SLAs). Sturm et al. (2000, p. 13) define the term SLA as ”a contract between IT and its clients that specifies the parameters of system capacity, network performance, and overall response time required to meet business objectives. The SLA also specifies a process for measuring and reporting the quality of service provided by IT, and it describes compensation due [for] the client if IT misses the mark”. Service quality is a key aspect of an SLA. According to Berger (2012), the service quality is defined by the following elements of the SLA:

- *Service level parameters* (SLPs) represent certain quality characteristics of a service that are to be measured.
- *Service level objectives* (SLOs) are the target values of the SLPs that need to be met.
- *Measurement methods* are used to determine the way the SLOs are measured as well as the frequency and the precision of measurement.

Because cloud labor is a service that is delivered via IT, the concept of the SLA can also be applied to cloud labor.

3.1.2. Relevant quality dimensions

Parasuraman et al. (1985) have identified ten ”determinants of service quality” which are summarized in table 3.1 and which are used as a starting point to further elaborate on the relevant quality characteristics of cloud labor services. For cloud labor services, these

Table 3.1.: ”Determinants of service quality” adopted from Parasuraman et al. (1985).

Characteristic	**Description**
Reliability	Consistency, dependability, accuracy, correctness, timely delivery of results
Responsiveness	Willingness or readiness of employees to provide service
Competence	Skills and knowledge to perform the service
Access	Approachability and ease of contact
Courtesy	Politeness, respect, consideration, and friendliness of contact personnel
Communication	Keeping customers informed in a language they can understand and listening to them
Credibility	Trustworthiness, believability, honesty
Security	Freedom from danger, risk, or doubt
Understanding	Making the effort to understand the customer’s needs
Tangibles	Physical evidence of the service

quality characteristics are relevant in two ways, from a technical perspective and from a workforce perspective.

Technical aspects of quality are covered in the characteristics *reliability*, *access*, *security* and *tangibles*. They can be mapped to the common quality characteristics of IT services.

Examples for the characteristic *tangibles* would be the server infrastructure of the platform as well as the technical infrastructure of the service workers. The platform infrastructure needs to meet specific performance, scalability and availability requirements in order to guarantee a continuous service with high performance. For the service worker, a certain minimum network bandwidth, CPU speed and a high quality display might be required in order to work on data intensive tasks with complex images or diagrams to be revised.

From a workforce perspective, especially those characteristics that assume the presence of humans (e.g. responsiveness, competence, courtesy, communication) can be considered highly relevant for the service workers but also for the service personnel of the platform.

However, for the cloud labor service itself, which is provided through an abstract interface, the requester is primarily interested in *what* is returned and *when* it is returned. The *what* refers to the accuracy of the information being returned, the *when* refers to the response time of the service and also to its availability and scalability. With regard to the quality characteristics listed in table 3.1, accuracy and response time are covered by the *reliability* characteristic while availability and scalability are covered by the *access* characteristic. Accuracy, response time, availability and scalability are discussed in the following:

Accuracy

For cloud labor services, accuracy refers to the quality of the results with regard to the information that the cloud labor platform delivers back to the requester. As mentioned previously, among all quality dimensions, the by far largest amount of the quality related research articles in the area of cloud labor services deal with accuracy. Often, the broader term *quality* is used as a substitute for accuracy. The QM challenges mentioned in section 2.3.2 predominantly refer to the aspect of accuracy. This is mainly because of the anonymity of the workers and the limited control that the requesters and the platform have over them. Other aspects of quality are usually much less relevant as long as the requester cannot rely on the accuracy of the results.

The actual requirements with regard to accuracy may be manifold. They can be divided into 16 dimensions of information quality that have been identified by Kahn et al. (2002): *Accessibility, appropriate amount of information, believability, completeness, concise representation, consistent representation, ease of manipulation, free-of-error, interpretability, objectivity, relevancy, reputation, security, timeliness, understandability and value-add.* When defining a work request, the requester needs to reflect which of the quality dimensions are important to him and needs to explicitly specify the appropriate requirements. The role of the QM mechanism is to ensure that the quality requirements are met. The accuracy of a result is usually rated in a binary way: it is considered *correct* if the requester's requirements are met and is considered *incorrect* if they are not. Sometimes, results are also *graded* according to a predefined scale (Hoffmann, 2009; Little et al., 2010a; Liem et al., 2011).

Availability and scalability

Availability can generally be defined as the "ability of a[n] ... IT Service to perform its agreed function when required" (Lloyd & Rudd, 2007, p. 290). Correspondingly, the availability of cloud labor services can be defined as the ability of delivering a cloud labor service with an agreed scope and volume whenever required. This implies a contract in which the scope and the volume are defined. While the technical availability requirements can be addressed by means of IT service management, the workforce requirements pose a challenge for the concept of cloud labor services. Especially in the case of a public workforce, workers cannot be forced to work on specific tasks, but they must be motivated to do so. Because many requesters and platforms are competing for the same workers, the availability of the workers and their demand for a certain type of task may change over time. In fact, during the research work on this thesis it was observed that tasks which once had been well received by the workers were not picked up at all at another point in time. Moreover, workers may agree not to work on certain tasks because they dispute the honesty of the requester or are unhappy with the payment. Increasing the compensation may put those workers back to work. Cultivating a good relation to the workforce and reacting to their inquiries is also important, as shown in section 2.6.2.

Scalability can be defined as the "ability of an IT service, process, configuration item, etc. to perform its agreed function when the workload or scope changes" (Lloyd & Rudd, 2007, p. 308). Accordingly, the scalability of cloud labor services can be defined as the degree to which a cloud labor service is able to perform its agreed function when the workload or scope changes. The more scalable the service is, the smaller is the increase in cost if the demand grows and the higher is the decrease in cost if the demand declines. Because of the pay-as-you-go nature of cloud labor services, the requester normally has to pay only for the work that is actually being performed, which results in a very good scalability. However, there are recent experiments to reserve workers by paying them for remaining stand-by waiting for tasks (Bernstein et al., 2011). Even though these approaches are targeted to improve the response time of tasks rather than to manage the scalability, similar approaches may emerge for the latter objective.

Because of the on-demand nature of cloud labor services, availability and scalability represent inherent strengths of the concept and can be seen as the most remarkable quality characteristic of cloud labor services. Apparently, an active workforce management is essential in order to ensure the availability and scalability of cloud labor services. The progress of the task execution needs to be tracked continuously and corrective actions need to be taken if the tasks are not picked up in time by appropriately skilled workers.

Response time

The response time of a cloud labor service is the period of time between submitting a task to a cloud labor platform and receiving the result. The response time includes the time workers need for finding and executing the task, the computing time for processing the tasks and the responses, and the technical communication overhead. In anticipation of section 3.3, multiple workers may be involved until a reliable result is delivered back to the requester which may increase the overall response time of the task.

The execution time of an individual task is influenced by a series of factors including the nature of the tasks and the design of the user interface. Compared to IT services, the response time of cloud labor services is usually very high because of the humans being involved. However, if there are many equivalent tasks that may be performed in parallel, the overall response time can still be relatively small, provided that enough workers with the required skills are available. A batch oriented task like validating a 40 page product catalog or transcribing a 60 minute audio recording may be completed in minutes if enough workers are working in parallel. Therefore, the overall completion time will not just depend on the execution time of the individual tasks, but also on the scalability of the service.

From the workforce perspective, the overall response time may be reduced in three ways:

- Reducing the time workers need for finding the task.
- Reducing the time for executing the tasks.
- Parallelizing efforts.

There are recent approaches in the area of *real time crowdsourcing* that address these aspects in different ways. They will be discussed in section 3.2.4.

3.1.3. Influencing factors

This section describes what factors are influencing the quality of the results delivered by the cloud labor platform. It is structured along the three perspectives of the basic cloud labor concept: Application, platform and workforce.

Application perspective

Naturally, a proper description of the task objectives and the task specific capabilities of the workers have a severe impact on the quality. Examples for correct responses should be provided and the workers should have the option to contact the requester if they have questions regarding the task (Kaufmann et al., 2011). This will not only help to avoid misunderstandings regarding the task objectives, but also to anticipate and avoid similar questions from other workers in the future as it allows for adjusting the task description.

Huang et al. (2010) have observed that a fine grained task granularity can improve the effectiveness of a task by generating more high quality results. They have developed a model that can predict the impact of the task granularity on the accuracy. In their experiments, the task granularity is represented by the number of images to be labeled and the number of labels to be returned. In general, as discussed in section 2.6.4, tasks should be designed in a way that they target a suitable type of workers.

Platform perspective

The platform influences the quality of the cloud labor services in three ways: By its technical characteristics, by its ergonomics and by the features it provides.

Like for any other IT based system, the reliability of the technical infrastructure is naturally a fundamental prerequisite for providing high quality cloud labor services. If the platform breaks down or if it cannot satisfy the workload in a timely manner, the remaining considerations discussed in this chapter are becoming irrelevant.

The ergonomics of the platform is another crucial point. If the platform is not attractive and easy to use, workers and requesters may choose a different platform, which reduces the availability of skilled workers and the variety of available tasks. In addition, ambiguous controls or faulty usage instructions may affect the accuracy of the work results and also the response time.

The platform features can be seen as an enabler for other quality relevant factors. The lack of a feature may make it impossible or at least difficult to use certain options. For example, if there is no support for workers to get in touch with the requesters, it will be difficult for them to provide feedback about blurry task instructions. Section 2.3.2 provides a summary of key platform requirements. A particularly important feature is the existence of appropriate QM mechanisms, which will be discussed in the remainder of this chapter. However, such mechanisms do not necessarily have to be part of the platform itself but can be implemented by the requester on the basis of fundamental platform capabilities. An example for such capabilities can be found in section 5.2.1.

The service model of the platform is crucial because the size and the capabilities of the available workforce will naturally have a key impact on all aspects of quality. If the required skills are not available, the tasks may not be accurately completed or the response time may increase because less skilled workers are likely to need more time to complete a task. If there are too few adequately skilled workers, the availability and the scalability of the cloud labor service may suffer. For location based services, a geographically distributed workforce is a key factor. Meta-platforms like crowdflower.com that merge workforces from multiple platforms may establish access to a much larger and more uniformly distributed workforce with a greater variety of skills than independent ones.

Workforce perspective

In the long run, the skill level as well as the number of available workers are both influenced by the task design and by the way the workers are trained and motivated. This includes effective and sustainable education and feedback mechanisms as well as an adequate compensation of the workers.

While several studies consistently report that increasing the *payment* does not increase the accuracy of the results, a finding that is consistent with the standard economic theory, ”higher payment leads to quicker results” because it ”attract[s] workers at higher rate” who then also completed more tasks; simplifying the tasks had a similar effect (Mason & Watts, 2010; Rogstadius et al., 2011).

Faridani et al. (2011) have presented a model for predicting completion times depending on the price. They point out that the form of payment can also have an impact. In their experiment, workers had to find words in a word puzzle. Payment per completed puzzle (”quota” system) turned out to be more effective than payment per word (”piece rate” system). The quota system lead to ”better work for less pay” than the piece rate system.

Accuracy has been observed to be significantly increased though intrinsic motivators. In an experiment performed by Rogstadius et al. (2011), workers were asked to identify pictures that showed blood cells infected with malaria parasites. The exact same task was used in two setups: Once as a charity task (with no compensation) under the name of a non-profit organization and once as a paid task under the name of a major actor in private pharmaceutical manufacturing. The accuracy turned out to be some 10% higher in the non-profit setup. Furthermore, the non-profit setup has attracted more workers who each completed a larger number of tasks.

3.2. Quality management approaches

A number of general QM approaches for cloud labor services have been described in the scientific literature which are outlined in the following sections.

3.2.1. Qualification tests

Qualification tests are commonly used for assessing the qualification of workers before they are allowed to work on a new type of task. In such a test, the worker is asked to respond to a series of test questions which are representative of the real tasks that make up the application. Depending on the type of application, the test is either evaluated manually or it is assessed automatically by comparing the worker responses with the already known optimal responses. Depending on the portion of test questions the worker has answered correctly, a failure rate is assigned to the worker which can be used in several ways. On the one hand, it can be used as an entry barrier to ban spammers and unskilled workers from the task, i.e. workers who exceed a certain minimum error rate will not be allowed to work on the tasks. On the other hand, the worker failure rates may be used to initialize the parameters of advanced QM mechanisms. Qualification tests represent a fundamental feature of many cloud labor platforms including MTurk and CrowdFlower. A useful side-effect of qualification tests is that they can also be used for training purposes. Given that the workers have to complete the test *before* they start working, they also have to complete the training.

3.2.2. Output-based quality management

Most existing approaches for managing the accuracy of cloud labor services are *output-based* as they solely look at the responses received from the worker in order to assess their quality. Among these approaches, again the vast majority are *crowd-based* as they are outsourcing the QM effort back to the crowd. According to different patterns, contributions of multiple workers are aggregated or iteratively transformed in order to deduce a reliable result. Another important approach is *gold-based quality management* which relies on tasks for which the optimal response is already available upfront. By randomly passing such tasks to workers and comparing their responses with the optimal ones, the average accuracy of the responses can be estimated. The most relevant patterns for output-based QM will be introduced in section 3.3.

3.2.3. Execution process monitoring

Execution process monitoring does not measure the accuracy of worker responses but aims to gather insights about the way a worker is performing a task by monitoring, tracking or analyzing the actual execution process. Such approaches are primarily used for human-directed cloud labor. The most prominent example is probably ODesk's *Work Diary* that screens the workers' progress by taking regular screenshots on their computers and making it available to the requesters.[1] The tool is mainly supposed to support ODesk's pay per hour model by documenting that a worker is actually working on a specific project. That way, unreliable workers can be detected.

Similar approaches have been recently applied to cloud labor services for collecting information about the worker behavior and deriving predictions about the response quality. Using a JavaScript library, Rzeszotarski & Kittur (2011) are monitoring the activities a worker performs on the task interface while executing a task, including mouse movement, mouse clicks and key strokes. Based on a "task fingerprint" generated from that data, the authors were able to estimate the accuracy of the worker's responses. Their "CrowdSwape" tool combines this approach with complementary QM approaches like crowd-based and gold-based QM (Rzeszotarski & Kittur, 2012). By merging the predictions and insights gathered by the different approaches through machine learning and by visualizing them in an interactive way, CrowdSwape aims to support requesters in better exploiting the crowd.

3.2.4. Response time management

There are many applications in which the response time of the cloud labor services matters. This is specifically the case for synchronous applications in which the requester is waiting for a response or in which a dynamic process is supposed to be supported by cloud labor services in real-time. An example for the first category is VizViz, a mobile phone application for blind people that answers visual questions in nearly realtime (Bigham et al., 2010). Using the application, a requester can take a picture and record a spoken question

[1] https://kb.odesk.com/categories/Clients/Managing+My+Team/Work+Diary/, last accessed on 2013-01-07.

regarding the picture, e.g. the request to read what is written on a sign. The picture and the voice message are transferred to the server component of VizViz, which publishes a corresponding task on the MTurk platform. A spoken response is provided by a worker and returned to the requester within about 30 seconds. That way, VizViz provides a low cost solution that helps blind people to interpret visual information. It is available on iOS and Android. The price for answering questions is about 4-5 cents. The server component is implemented using QuickTurkit[2], an abstraction layer on top of Turkit.

AudioWiz provides a similar service for transcribing speech recordings into text in near real-time (White, 2010). Another example for a near real-time application is Soylent (Bernstein et al., 2010), which has already been mentioned in section 2.4.3 and will be further discussed in section 3.3.6.

As outlined in section 3.1.2, the response time depends on the delay until a worker starts working on a task, on the execution time as well as on the degree of parallelization. Bernstein et al. (2011) introduce the term of *synchronous crowds* that support real-time crowdsourcing. They propose a combination of two models in order to reduce the response time: The *retainer model* reduces the time until a worker starts working on a task by paying a small compensation for being on standby. That way, workers are at hand and start working immediately once a task arrives. The *rapid refinement* model reduces the execution time. It helps the workers to deliver a response more quickly by providing them with real-time feedback about the progress of other workers who are working on the same task in parallel. The underlying idea is based on the observation that many workers ”decide on the gist of a solution quickly, but can take time to commit the final answer”. This gist is being identified and used to narrow down the search space for all workers so they can get to a solution more quickly. In their *Adrenaline* application, Bernstein et al. (2011) use the rapid refinement model in order to identify the best frame out of a short video sequence. They have shown that the approach can drastically reduce the execution time while still achieving a reasonable result quality.

An example for a dynamic process supported by cloud labor services is Legion (Lasecki et al., 2011), which allows for real-time control of GUI applications by the crowd. It captures the screen of existing GUI applications and makes them available to crowd workers using MTurk. In an example application the authors demonstrate how the system can be used to collaboratively control a robot toy, a task that naturally needs to be performed in real-time. Different mediation strategies are evaluated to aggregate feedback from multiple workers. A promising strategy turned out to be the ”leader” strategy in which direct control is temporarily passed to a single worker who did agree well with the others in the past. That way, the latency is reduced because there is no need to wait for a consolidated decision of multiple workers. At the same time, the risk of undesired operations is increased. Therefore, the optimal mediation strategy will depend on the application.

The following section will tie in with section 3.2.2 by focusing on output-based QM in more detail.

3.3. Patterns for output-based quality management

This chapter introduces the five most relevant approaches for output-based QM. They are organized into five patterns in this thesis. While the first pattern uses gold standard

[2] `http://quikturkit.googlecode.com.`, last accessed 2013-05-01.

tasks for estimating the performance of the workers, the remaining four patterns are crowd-based, i.e. they leverage the crowd itself for managing the accuracy of the worker responses.

Section 3.3.1 firstly introduces a set of task characteristics that are needed when discussing the reach of the individual patterns. Section 3.3.2 then describes the gold-based QM pattern, while the crowd-based patterns are covered by sections 3.3.3 through 3.3.6.

3.3.1. Relevant task characteristics

Not all output-based QM patterns can be applied to all tasks. The following classification aims to provide a structure for indicating the reach of the QM mechanisms described in the subsequent sections. The decision tree in section 3.4 will build on this classification. It consists of four binary dimensions which are outlined in table 3.2 and described in detail in the following.

Table 3.2.: Classification of cloud labor tasks as a basis for choosing a suitable QM approach.

Characteristic	**Value**	**Description**
Determinacy of task results	deterministic	There is exactly one well defined optimal response.
	non-deterministic	There may be multiple responses which perfectly meet the task requirements.
Validation effort	low	Validating a response requires much less effort than generating it.
	high	Validating a response requires at least similar effort as generating it.
Granularity	coarse-grained	Task could be divided into smaller tasks.
	fine-grained	Task cannot be further divided into smaller ones.
Difficulty	simple	A worker who met the qualification test can be expected to complete the task successfully.
	difficult	Even a worker who met the qualification test cannot be expected to complete the task successfully.

Note that the taxonomy naturally differentiates between distinct classes even though the underlying characteristics may represent a continuum. The following descriptions advise on how to determine the corresponding values:

- *Determinacy of task results:* Some quality management approaches use redundancy for ensuring the accuracy of worker responses. The exact same task is passed to independent workers and their responses are then aggregated. The first dimension of the classification refers to the question whether an automatic aggregation of the results is possible at all. This is the case if there is a well defined optimal response for the task i.e. if two workers who independently work on the task and who perfectly meet the task objectives return identical responses. In this thesis, such tasks are defined as *deterministic*. An example is the transcription of a speech recording if spelling and punctuation does not matter. For *non-deterministic* tasks, there may be multiple optimal results that differ from each other even though they all perfectly meet the task objectives. Examples include text authoring, language translation and the creation of creative designs.

- *Validation effort:* Another fundamental concept is the validation of worker responses by other workers. Its suitability obviously depends on the validation effort, which represents the second dimension of the classification. Depending on the type of task, the effort may be much lower than the effort for producing the original response or it may be of at least similar magnitude. For instance, in a text authoring application, it is usually much simpler to decide whether a given text meets certain quality criteria than to write the text. Another example are image research applications. It is much harder to find a picture which shows exactly five bananas than to validate that there are indeed five bananas on a given picture. Examples for a similar effort are basic classification and research tasks for which a result validation is only possible by performing the research again. The magnitude of the validation effort is assessed in a binary way: It is considered to be *low* if it clearly falls below the execution effort whereas it is considered *high* if it is of the same or higher magnitude.
- *Granularity:* Even small tasks may be split into even smaller parts. For example, instead of asking one worker to transcribe a 100 word handwritten text, 100 workers could be asked to transcribe just a single word. Obviously, such a decomposition does not always make sense because it increases the overhead for claiming the tasks and it hides context from the workers which may prevent them from delivering a good result. If an average worker is able to complete the entire task in a reasonable amount of time, a decomposition of the task is usually not recommended. However, the granularity of the task can make a difference from a quality management point of view. Therefore, it is used as the third dimension of the classification. A task is defined to be *fine-grained* if it could (theoretically) be divided into several smaller tasks, whereas it is *coarse granular* if this is not the case.
- *Difficulty:* The fourth dimension of the classification concerns the difficulty of the task. A *simple* task is defined as one that can be expected to be successfully completed by an average worker. *Average* refers to the specific crowd that is permitted to work on the task, i.e. even an expert may be considered an average worker if the specific crowd consists of experts only. Therefore, diagnosing an X-ray photograph of a common injury may be considered a simple task for a crowd of radiologists. However, diagnosing a very rare injury may be a *difficult* task even for an experienced radiologist. Maybe, only a small portion of the crowd would be able to successfully complete it. Breaking down the task into smaller parts would not work because diagnosing an X-ray photography is a holistic task that is already coarse granular according to the definition above. However, a task may also be difficult *because* it is fine-grained. Consider for example a ten page language translation with extremely high quality needs. If the requester is intolerant to mistakes, it will be very hard for a single translator to deliver a satisfactory result. Therefore, a second or even third worker should be assigned for proofreading which suggests that the difficulty of a task may have an impact on the selection of a suitable QM approach.

3.3.2. Gold pattern

Gold standard tasks are such tasks for which the correct response is already available. By comparing a worker's response with this *ground truth*[3], the accuracy of the response can be determined. This only works for deterministic tasks as defined in the previous section because for non-deterministic tasks a ground truth is not defined. Gold standard tasks are

[3] This term is sometimes used in the context of cloud labor services, e.g. by Sorokin & Forsyth (2008).

widely used for estimating the initial failure rates of workers or for assessing the quality of a stream of worker responses. Gold standard tasks are typically manually collected, which causes an extensive effort (Le et al., 2010; Oleson et al., 2011).

A basic but effective way of determining the accuracy of responses that are returned by a worker for a stream of tasks is to inject gold standard tasks into that stream (Sorokin & Forsyth, 2008). By comparing the incoming responses with the gold standard, samples may be taken from the stream of incoming results in order to determine the overall quality of the stream. Workers who fall below a minimum quality limit may be banned from the task in order to keep a high overall quality of the responses returned by all workers.

For non-deterministic tasks, the gold pattern can only be used indirectly by combining them with deterministic tasks, i.e. by putting verifiable (deterministic) questions into the non-deterministic task. Kittur et al. (2008) have shown that this approach can significantly improve the quality of the results. Naturally, it also increases the completion time because responding to the additional questions requires more time. However, apart from consistency checks, for subjective tasks like surveys and polls it is often the only way to assess the quality of the responses.

According to Le et al. (2010), an advantage of the gold pattern is that it may be used for implicit training of the workers in a dynamic learning environment. Given the fact that the correct responses are already available, instant feedback can be provided to the worker right after the results have been submitted. In case of incorrect responses the feedback may be complemented with predefined explanations regarding what has been wrong and how the error may be avoided in the future. The process described by Le et al. starts with an initial set of training tasks (”training period”) that is passed to workers who work on a certain task type for the first time followed by a sporadic injection of additional training tasks throughout the entire process. If the percentage of incorrect responses exceeds a certain limit, the workers are considered to be spammers and will not be allowed to work on the task any more. The authors point out that the distribution of the correct responses in the training period should match the distribution of the correct responses in the actual tasks because workers seem to develop a notion of that distribution. For example, if in a binary task most of the training tasks had to be answered with ”yes”, the workers will also answer most of the actual tasks with ”yes”.

A general challenge of the gold pattern is that workers may identify the gold standard tasks when being assigned to the same one multiple times. Therefore, the number of gold standard tasks must scale with the number of actual tasks. Oleson et al. (2011) use about one percent of gold standard tasks. In order to reduce the effort for the manual creation of gold standard tasks, they have developed a semi-manual approach for generating new gold standard tasks. Based on a set of known correct responses that have been gathered using the voting pattern to be described in section 3.3.3, their *programmatic gold* approach mutates the correct responses into incorrect ones according to a set of typical worker errors that they have manually identified upfront. Then, it merges correct responses with the various types of worker errors according to the distribution of those errors in the actual set of tasks. Because there is still a risk that in the voting phase all workers have agreed on an incorrect response, Oleson et al. provide a way to ”contest” gold standard tasks if a worker thinks that the gold standard task has been setup incorrectly. Once a certain amount of workers have contested the task, it will not be used any more. Even though there is still manual effort required, Oleson et al. report that their approach leads to significant time savings.

In general, it needs to be considered that the gold pattern cannot directly improve the accuracy of individual results to be passed to the requester because the results of gold standard tasks are already available *before* the worker performs a task. Instead, gold standard tasks can only be used to indirectly improve the accuracy of a stream of results by tracking the average performance of a worker. When a worker misses too many of the gold standard responses, corrective actions need to be taken, e.g. by educating the worker or in the worst case, by banning him from the task.

3.3.3. Voting pattern

In the *voting* or *redundancy* pattern, the exact same task is passed to a set of workers who are assumed to complete it independently of each other. The worker responses are then compared or aggregated in order to generate a reliable result. Figure 3.1 illustrates the basic flow of the voting pattern.

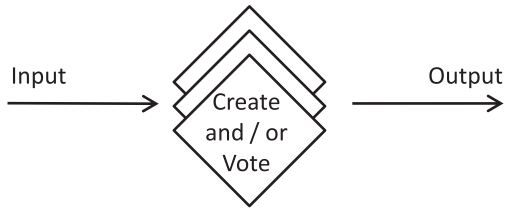


Figure 3.1.: Basic flow of the voting pattern.

The voting pattern is grounded on Condorcet's Jury Theorem (de Caritat marquis de Condorcet, 1785), which states that if there are two possible outcomes for a task (correct vs. incorrect) and each decision maker has the independent probability of more than fifty percent to make the right decision, the probability for a correct group decision is higher than the one of the individual worker. Variations of the voting pattern have been used in other domains, e.g. in medicine (Zhou et al., 2002) and machine learning (Sheng et al., 2008).

In the context of cloud labor services, various implementations of the voting pattern have been proposed, for example by Sorokin & Forsyth (2008), Snow et al. (2008), von Ahn & Dabbish (2008), Whitehill et al. (2009) and Ipeirotis et al. (2010). In its most basic form, the *majority vote*, the response returned by the simple majority of the workers is considered to be the correct one while all other responses are considered to be incorrect. By tracking the individual failure rates of the workers, spammers can be identified and banned from working on further tasks. Snow et al. (2008) have investigated how a group of non-experts compares to a single expert. For different text analysis applications, they have observed that depending on the concrete application a vote of two to ten non-experts is required to exceed the accuracy of a single expert.

As a general restriction, like the gold pattern, the voting pattern is limited to deterministic tasks. The *create/vote* step in figure 3.1 may represent either a multiple choice decision between a number of predefined options or it can represent the creation of a free-form response (for example a word being typed in) for which the worker implicitly votes by submitting it. In any case, there must be a high probability that multiple workers propose exactly the same response.

Furthermore, it has to be considered that agreement between the workers does not necessarily indicate a correct response, but may also be caused by a cultural bias, by ambiguity of the task or by cheating and collusion (Le et al., 2010). Another reason may be a lack of qualification. For example, two novices who do not understand the challenge of a task may overrule an experienced worker by choosing the apparently correct but actually incorrect response. Therefore, more sophisticated implementations of the voting pattern have been developed that take into account the individual qualification of the participating workers.

Based on the *expectation maximization* (EM) algorithm developed by Dempster et al. (1977) and inspired by the work of Dawid & Skene (1979), several authors (Raykar et al., 2009; Whitehill et al., 2009; Raykar et al., 2010; Welinder & Perona, 2010) have proposed a *maximum likelihood estimation* (MLE) that estimates both the worker failure rates as well as the ground truth at the same time. The basic idea is to look at a matrix of responses returned from multiple workers for multiple tasks at once and to find an estimate for the ground truth that keeps the average error rate of the involved workers at a minimum. The EM algorithm iteratively establishes a ground truth, calculates the corresponding worker failure rates and refines the gold standard in order to find a better match. If there is previous information available about the worker failure rates, it may be used as an input to the EM algorithm. The EM algorithm will formally be described in section 6.2.2.

There are situations in which workers are actually returning good results, but the results are biased in a certain direction. For example, in a website labeling application, Ipeirotis et al. (2010) have observed that mothers of young children tend to be more restrictive when deciding whether a website is suitable for children or not. In order to leverage the results returned by such biased workers, Ipeirotis et al. have extended the EM approach in order to separate the unrecoverable error rate from bias. Their experiments have shown that the approach is capable of identifying even ”sophisticated” spammers and, therefore, drastically reduces the execution cost.

The voting pattern may also be used for fine-grained tasks, provided that the individual ”grains” are not *difficult* to solve, i.e. that the task could be split into smaller ones which can be expected to be successfully solved by an average worker. An example for such a concept with only two participants is the ESP game described in section 2.2.3. Two players are asked to assign labels to the same image. As soon as a certain number of identical labels have been submitted by both workers, those labels are being accepted. Like the simple majority vote, the ESP game does not track the worker performance.

Altogether, the voting pattern has proven to be an effective way of generating high quality crowdsourcing results for deterministic tasks. Cloud labor platforms like MTurk and Crowdflower provide basic support for the voting pattern out of the box. However, in its simplest form as the majority vote, it multiplies the execution effort and, therefore, leads to high costs (Ipeirotis et al., 2010). Use of the EM algorithm can drastically reduce the execution effort while increasing the accuracy of the results, but that advantage comes with a burden because the EM algorithm turns the voting pattern into a batch process. In order to be efficient, workers must submit responses for maybe at least 100 tasks before the results can be identified and made available to the requesters and before any performance feedback can be provided back to the workers. This has a number of disadvantages: First, the approach is not suitable for real time applications in which a result needs to be returned to a requester more or less immediately. Second, as mentioned in section 2.6.3, delayed feedback will have less effect on the future worker performance. And even worse, if there had been misunderstandings about the task requirements, a large amount of work would have to be rejected which according to Oleson et al. (2011) may reduce the worker satisfaction and lead to extensive discussions and e-mail exchange with the worker.

3.3.4. Validation pattern

In the *validation* pattern, a task is passed to a single worker (*producer*) who returns a response. This response is then being passed to one or multiple additional workers (*reviewers*) who provide a binary rating whether to accept or reject the response. The validation pattern does not improve the quality of a result, but only acts as a filter which is supposed to filter out incorrect responses. When combining it with a feedback loop, the approach can be used to improve the skills of the producer. Figure 3.2 depicts the basic flow of the validation pattern.

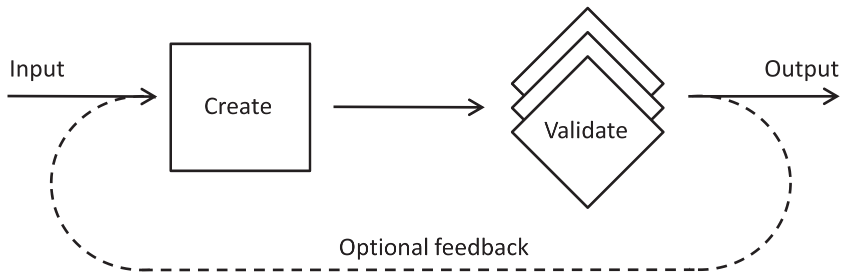


Figure 3.2.: Basic flow of the validation pattern.

The validation pattern is more generic than the gold or the voting pattern as it is not limited to deterministic tasks, but also works for non-deterministic tasks. Moreover, there is no need for a task specific aggregation or comparison of worker responses as it is the case for the voting pattern. Therefore, the validation pattern can easily be implemented for basically any type of task without setting up any task specific data processing apart from the user interfaces for gathering the reviewers' decisions and feedback and presenting it to the producers. Little et al.'s (2010a) *rating tasks* are broader than validation tasks. Instead of asking a binary question whether the task objectives are met or not, grading tasks ask to what extent the objectives are met on a given scale, for example by a number between 1 and 10.

Hirth et al. (2013) describe a *control group* approach in which the producer's response is passed to a fixed number of reviewers. If the majority of the reviewers confirm the correctness of the response it is accepted otherwise it is rejected. The reviewers are always paid, the producer is only paid if the response has been accepted. The authors have shown on a theoretical basis that their approach identifies spammers as good as the majority vote approach. Based on a model for calculating the expected costs of both approaches they could show that for high priced tasks, the control group approach outperforms the majority vote approach while it is the other way round for low priced tasks. The cost model can also be used to calculate the optimal trade-off between quality and cost by taking the cost for incorrect results into account. The authors have further observed that the control group approach is much more cost-efficient when using better workers. This is not surprising, as the approach does not take the performance of the individual workers into account and thus does not provide any means for banning spammers or bad performers.

The validation pattern seems to be rather rarely used as it is only mentioned in a small number of articles including the ones of Sorokin & Forsyth (2008) and Hirth et al. (2013). Tools like TurKit (Little et al., 2010b), CrowdLang (Minder & Bernstein, 2011), CrowdForge (Kittur et al., 2011) or Crowdweaver (Kittur et al., 2012) described in section 2.4.3 support the implementation of the validation pattern out of the box.

3.3.5. Iteration pattern

The *iteration* pattern can be seen as an umbrella for such approaches in which a response delivered by a producer is reviewed and possibly improved by a single reviewer or a chain of reviewers until it meets the requester's requirements.

In the implementation used by the speech-to-text transcription service CastingWords[4], there is an additional grading step that is performed by the reviewers before working on the actual improvement. According to Hoffmann (2009) and Liem et al. (2011), the process works as follows: Audio recordings are split into overlapping three to four minute segments and are passed to the MTurk platform for transcription. Workers are asked to listen to a segment of the recording and transcribe it from scratch or to grade and improve a transcript created by another worker. That way, the segments are improved step by step until they are reassembled to a full draft which is again graded and polished by workers before it is finally sent back to the requester. Depending on the required turnaround time, the requester has to pay between \$0.70 and \$2.50 per minute of the audio. The grading steps are used to estimate the transcription effort of the workers and to determine their compensation. In addition, they are used to track the gradual improvement of the transcripts and to determine when it meets the requester's needs. If the grading step is performed by the same worker who improves the transcript, workers may tend to underestimate the quality of the original transcript in order to let their own contribution appear more valuable. Therefore, CastingWords is spending additional effort on grading the graders (Hoffmann, 2009; Liem et al., 2011).

Little et al. (2009a) have proposed a generic form of an iteration pattern (TurKit[5]) in which the improvement step is followed by a voting step. The process is illustrated by figure 3.3. In the voting or *ballot* step, two voters independently compare the original response of the producer with the one that has been improved by the reviewer and decide which one better matches the task requirements. If they do not agree, an additional voter is asked. The winning result is fed back into the iteration process. The iteration is continued until some termination condition is met. The voting step is supposed to ensure that the improvements added by the reviewers are not trivial and do not compromise the previous iteration.

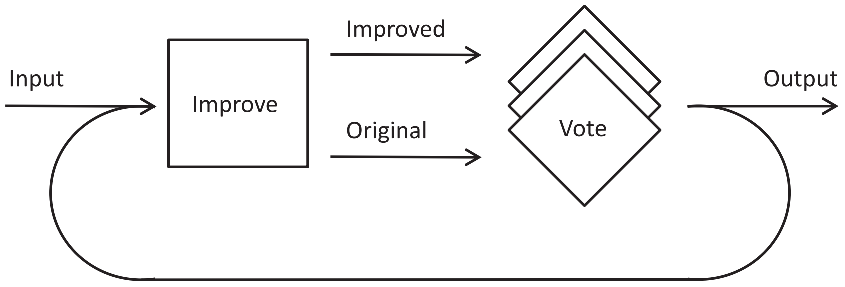


Figure 3.3.: Basic flow of the iteration pattern; adopted from Little et al. (2009a).

The iteration pattern has been observed to provide compelling results for writing, brainstorming and also transcription tasks, but it has the disadvantage that the initial response

[4] `http://castingwords.com/`, last accessed on October 11, 2013.

[5] Note that the authors have used the name *TurKit* for both the specific implementation of this iteration pattern as well as for the generic workflow environment described in section 2.4.3

may set the direction for the subsequent workers (Little et al., 2010a). Kulkarni et al. (2011) has also observed that an incorrect initial contribution can "derail" the task, i.e. it may set subsequent workers on the wrong track. Therefore, the iteration pattern should only be used if already the first worker has a realistic chance to deliver a reasonable response which according to the definition in section 3.3.1 is not the case for *difficult* tasks.

An important limitation of Little et al.'s implementation of the iteration pattern is that it does not take the worker performance into account. Dai et al. (2010, 2011) have addressed this point by developing *TurKontrol*, a decision theoretical framework for the iteration pattern that uses a *partially-observable Markov decision process* (POMDP) to model the three decision points of the approach. Based on the expected utilities of the appropriate actions, TurKontrol decides whether to add more voters to a voting step, which of the two responses to use as an input for the next iteration and when to stop the iteration process and to submit the final response. As input parameters, the model requires the expected utility of a result depending on its quality as well as the costs for an iteration and a ballot task. TurKontrol does not require any prior information about the worker failure rates but starts with initial worker-independent estimates. Those estimates are incrementally updated as the system gathers information about the performance of the individual workers (Dai et al., 2010, 2011).

Dai et al. (2011) have compared their adaptive TurKontrol with Little et al.'s (2009a) non-adaptive TurKit implementation of the iteration process. In an image description application, 655 workers have been asked to describe the scene being shown on a given picture. When spending the same amount of money, the quality of the results generated by TurKontrol outperformed the quality generated by TurKit by 11% in average. The author's have further shown that in order to reach the same quality with TurKit, 28.7% more money had to be spent. This is because the quality is not linear in the total cost, because the higher the quality of an iteration, the more difficult it is to improve it. Remarkably, the two mechanisms behaved differently in the ballot step of the experiment. While TurKit naturally always uses two or three voters in the ballot phase depending on whether the first two agree or not, TurKontrol did not assign any voters to the ballot for the first two iterations of each image, because it expected the utility of moving directly to a new iteration to be higher than to run a ballot step. The mechanism simply expected that during the first iterations, most workers would be able to improve the previous iteration. These observations also suggest that with TurKontrol, fewer voters are needed to achieve the same accuracy in the voting step.

In a language translation application, Kittur (2010) has observed that "seeding" a task by providing an initial solution as a starting point may improve the acceptance and the completion of the task. Therefore, it may be helpful to use a generated solution (e.g. a computer translation) as a starting point for the iteration pattern.

The iteration pattern can be modeled with tools like TurKit (Little et al., 2010b), CrowdLang (Minder & Bernstein, 2011), CrowdForge (Kittur et al., 2011) or Crowdweaver (Kittur et al., 2012) that have been described in section 2.4.3.

3.3.6. Comparison pattern

As discussed in the previous sections, neither the validation nor the iteration pattern is suitable for *difficult* tasks. In fact, Little et al. (2010a) have shown for a creative task that a parallel pattern gains better results than the iteration pattern. When asking workers to brainstorm company names they have observed that "the very best names seem to be generated by workers working alone". The voting pattern represents a parallel pattern but does not support creative tasks as they are typically non-deterministic. The parallel pattern proposed by Little et al. overcomes this limitation by adding a "comparison" step to the voting pattern that lets other workers identify the best response.

In this thesis, the pattern is called *comparison*. In a first step, a task is passed to a group of producers who return responses. The responses are then presented to one or more *voters* who decide which of the proposed solutions is the best. The voters may also decide that there are multiple equivalent solutions. Obviously, the producers who proposed responses in the redundancy step are not allowed to act as voters. The approach can be basically seen as a combination of two voting steps, the second of which is used to consolidate the responses of the first one. Figure 3.4 illustrates the basic flow of the voting pattern.

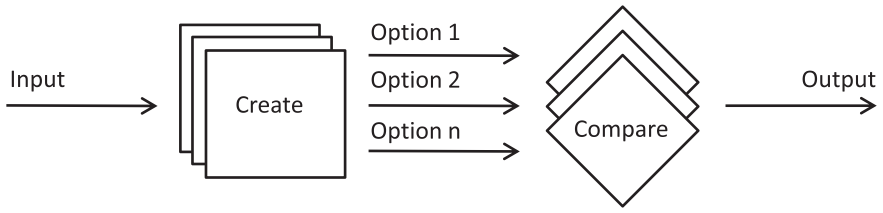


Figure 3.4.: Basic flow of the comparison pattern.

For many types of tasks, the voters will be able to identify the best solution, even if they would not have been able to provide it. For example, identifying a person shown on a picture is only possible if the worker at least has a guess, whereas verifying the result can be simply done by looking for another picture of the same person and comparing it with the given one.

A specific implementation of the comparison pattern is *find-fix-verify* (Bernstein et al., 2010) that has originally been designed for shortening or proofreading paragraphs of a given text. In find-fix-verify, there is an additional *find* step at the beginning in which multiple workers are asked to identify passages of the given text that should be reworked. If more than 20% of the workers agree that a specific passage needs rework, for each of those passages the fix-and-verify steps are initiated which together basically represent the comparison pattern. In the fix step, multiple producers are asked to provide recommendations on how to rework (shorten or revise depending on the actual application) the passage. In the vote step, the best solution is then identified by a group of voters. An implementation of the approach is available as a plugin for Microsoft Word$^{\mathrm{TM}}$. Under the cover it uses the TurKit engine (Little et al., 2010b), which accesses Amazon's MTurk cloud labor platform. In a series of experiments, find-fix-verify has successfully been used to shorten texts by 85% and to identify and correct 82% of spelling and grammatical errors (Bernstein et al., 2010).

In general, while the comparison pattern is specifically useful for difficult tasks, it is not recommended for tasks that are fine-grained at the same time. For instance, when transcribing a 5 min speech recording into text, even an expert will likely make a few mistakes. Therefore, *choosing* between alternative transcripts from different experts would likely not lead to a good result. The iteration pattern instead would provide the opportunity to *merge* the expertise of multiple workers, i.e. transcription errors made by one worker could be corrected by others.

3.4. Comparison of output-based approaches

In this section, the relevance of the individual output-based QM patterns and the rationale for using each of them are discussed (section 3.4.1) and the gaps of the individual approaches with regard to the objectives of the thesis are identified (section 3.4.2).

3.4.1. Decision matrix

This section summarizes the observations made when discussing the individual crowd-based quality management patterns in section 3.3 and classifies them according to the characteristics introduced in section 3.3.1.

Table 3.3 indicates which pattern can be applied to what category of tasks. The recommended approaches are printed in bold. In general, validation can be used for any simple task while iteration can be used for all fine grained tasks. The gold pattern can be applied to all deterministic tasks, whereas voting can be only applied to those deterministic tasks that are not also difficult and coarse grained at the same time. Finally, comparison can be used for any task except if it is difficult and also fine grained.

Table 3.3.: Decision matrix for choosing an adequate quality management pattern for cloud labor services.

	Deterministic			Non-deterministic		
Simple		Coarse grained	Fine grained		Coarse grained	Fine grained
	Validation effort low	Gold Voting **Validation** Comparison	Gold Voting Validation **Iteration** Comparison	Validation effort low	**Validation** Comparison	Validation **Iteration** Comparison
	Validation effort high	Gold **Voting** Validation Comparison	Gold Voting Validation **Iteration** Comparison	Validation effort high	Validation **Comparison**	Validation **Iteration** Comparison
Difficult		Coarse grained	Fine grained		Coarse grained	Fine grained
	Validation effort low	Gold **Comparison**	Gold Voting **Iteration**	Validation effort low	**Comparison**	**Iteration**
	Validation effort high	Gold **Comparison**	Gold Voting **Iteration**	Validation effort high	**Comparison**	**Iteration**

For several combinations of task characteristics, multiple patterns are competing which may differ in terms of efficiency. Figure 3.5 provides a decision tree that is intended to help to identify the presumably most efficient crowd-based pattern. By reading the diagram from top to bottom, the task characteristics can be assessed step by step and an adequate QM pattern can be identified.

The following paragraphs discuss for each of the patterns, why it is assumed to be of advantage compared to the possible alternatives:

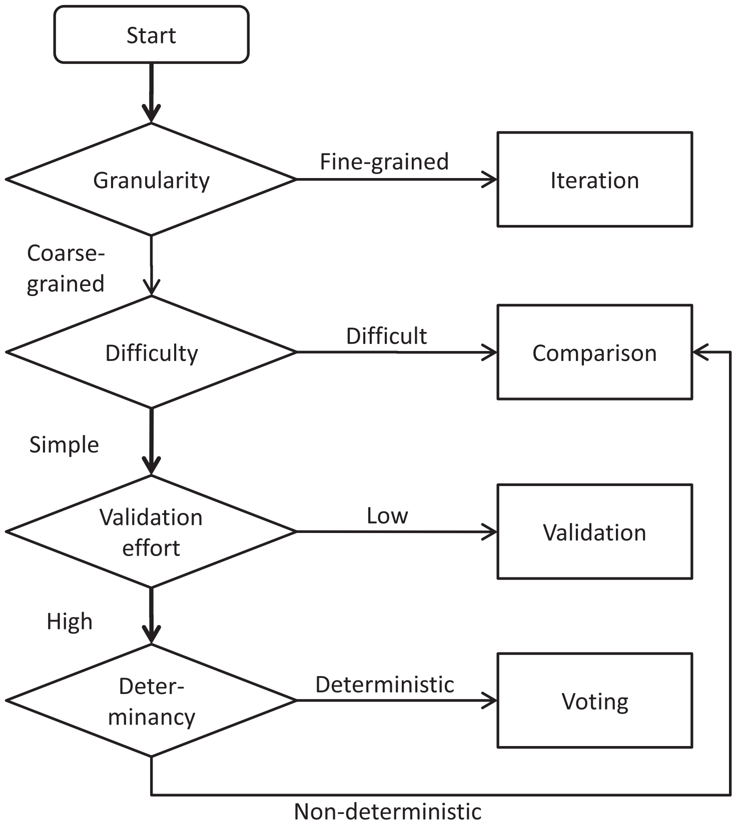


Figure 3.5.: Rule-of-thumb for choosing a suitable quality management pattern for cloud labor services.

- The *iteration pattern* is considered to be most efficient for all fine-grained tasks. This is because for fine-grained tasks there is a high probability that even experienced workers return at least one incorrect "grain". Using the iteration pattern, mistakes can be corrected by the subsequent workers without the need to perform the entire task again. All other patterns require redundant work. The voting pattern may still be a viable alternative as it allows for merging the results of several voters.
- The *comparison pattern* can be assumed to be most efficient for all coarse-grained, difficult tasks for which it is anyway the only option. In addition, it should be considered for simple, non-deterministic and coarse-grained tasks for which the validation effort is high. Depending on the application, the validation pattern could still be an alternative.
- The *validation pattern* is assumed to be most efficient for coarse-grained, simple tasks for which the validation effort is low. This is because the validation pattern gets likely along with a single execution of the task while the voting and comparison pattern would necessarily need multiple executions.
- The *voting pattern* should be considered for tasks that are coarse-grained, simple, deterministic and that cause a high validation effort. Because the majority of responses can be assumed to be correct ("simple task"), they can be directly aggregated in order to generate a reliable response. Additional validation or voting steps, as they are part of the voting and comparison patterns would cause an overhead ("high

review effort"). The voting pattern may also be used for fine-grained deterministic tasks. This will result in additional implementation effort for aggregating the individual "grains" but it can reduce the execution effort compared to the iteration pattern. An example for such a task will be discussed in section 6.1.

- The *gold* pattern may be considered for any deterministic task, if there is a limited number of tasks to be performed and a reasonable set of gold standard tasks is already available. Otherwise, it may be taken into account as a complement to crowd-based QM in order to reduce the risk for cheating.

The diagram in figure 3.5 should be considered as a rule-of-thumb and does not remove the need for considering other QM patterns according to table 3.3.

3.4.2. Gap analysis

This section analyzes the existing QM approaches for cloud labor services with regard to the research question to be addressed by this thesis. According to section 1, the objective is to investigate how the accuracy of cloud labor services can be managed in an efficient, scalable and goal-based way.

Table 3.4 summarizes the assessment of the existing QM approaches for each of those characteristics and compares them to the CSP/DVM[6] that will be introduced in chapter 5. A filled circle means that a characteristic is well supported, a half-full circle means that it is moderately supported, while an empty circle means that there is no or only weak support. Accordingly, the ratings *high*, *medium* and *low* are used.

- The *efficiency* rating of a crowd-based approach depends on how well the QM effort adjusts dynamically to the actual quality of the responses. This is currently only the case for TurKontrol and to some extent for TurKit[7]. For the basic gold pattern the efficiency rating depends on whether gold standard tasks are already available or have to be manually generated with high effort. For the programmatic gold approach, the crucial point is the efficiency of the crowd-based approach that is used to generate the gold standard tasks. All other approaches listed in the table result in a fixed QM effort because they always employ the same number of workers in order to come to a decision.

- The *scalability* rating is considered to be high for all crowd-based approaches, i.e. for all approaches except the ones belonging to the gold pattern. For the basic gold pattern, the scalability is considered low because even if an initial set of gold standard tasks is available, they will be exhausted at some point in time so manual effort is needed for generating additional ones. The programmatic gold approach is only partly crowd-based because it requires at least some manual effort. Therefore, a medium rating was chosen for it.

- The *ability to meet goals* is considered high for approaches that automatically control the accuracy of the results according to predefined quality objectives. This is actually not the case for any of the existing approaches. However, the gold-based approaches, the EM-based approaches and TurKontrol are at least capable of estimating worker failure rates which can be seen as a first step towards a goal-based

[6] The abbreviation stands for the combination of *continuous sampling plan* (CSP) and *dynamic voting mechanism* (DVM).

[7] Because TurKit uses a variable number of voters for the ballot tasks (either 2 or 3).

quality management. Therefore, the ability to meet goals is considered medium for these approaches. For the remaining approaches it is considered low.

Table 3.4.: Comparison of the output-based QM patterns with regard to the objectives of the thesis.

Pattern	Approach	Reference	Efficiency	Scalability	Ability to meet goals
Gold	Basic gold-based QM	(Le et al., 2010)	○/ ◐/ ●	○	◐
	”Programmatic gold”	(Oleson et al., 2011)	○/ ◐/ ●	◐	◐
Voting	Basic majority vote	(Snow et al., 2008)	○	●	○
	”Majority decision”	(Hirth et al., 2013)	○	●	○
	EM	(Raykar et al., 2009)	○	●	◐
	EM with bias detection	(Ipeirotis et al., 2010)	○	●	◐
	CSP/DVM	**(This thesis)**	●	●	●
Validation	”Control group”	(Hirth et al., 2013)	○	●	○
Iteration	”TurKit”	(Little et al., 2009a)	◐	●	○
	”TurKontrol”	(Dai et al., 2011)	●	●	◐
Comparison	”Parallel pattern”	(Little et al., 2010a)	○	●	○
	”Find-fix-verify”	(Bernstein et al., 2010)	○	●	○

● ≡ high, ◐ ≡ medium, ○ ≡ low

The low rating for many of the approaches does not mean that they are not valuable. As discussed in section 3.4, depending on the application only a subset of the patterns may be applied anyway. In addition, it needs to be considered that all crowd-based QM patterns implicitly make use of the voting pattern. By using an efficient and goal-based implementation of the voting pattern, the efficiency of the other patterns may be increased and they may even be turned into goal-based approaches. However, such an implementation is obviously missing. The CSP/DVM presented in section 5 is supposed to close this gap by offering a QM approach that is *efficient*, *scalable* and *goal-based* at the same time. Before introducing the CSP/DVM, the following chapter will provide fundamental considerations of statistical quality control essential for the reader to understand the conceived value of the new quality management approach.

4. Statistical quality control

The general objective of statistical quality control (SQC) is to make a statement about the quality of multiple items in an efficient way by only assessing a sample of them. This chapter covers aspects of QM and SQC that are needed as a foundation for the actual contribution of the thesis. In particular, the acceptance sampling plans and continuous sampling plans will be used for developing the models in chapters 5 and 6. The description is limited to aspects relevant to understand the derivation of the newly proposed methodology.

Sections 4.1 and 4.2 provide an overview of the basic concepts of QM and SQC, while sections 4.3 and 4.4 introduce relevant aspects of acceptance sampling and continuous sampling plans.

4.1. Overview

This section provides an overview on the field of QM. The general areas of QM are introduced in section 4.1.1. Section 4.1.2 then focuses specifically on SQC.

4.1.1. Quality management

According to chapter 3, quality management (QM) is crucial for cloud labor services. In the following, the general areas of QM are introduced. A project centric view was chosen in order to put the concept into a managerial perspective. Following Rose (2005, p. 41), QM spans the entire process from quality planning, quality assurance and SQC[1] to quality improvement:

Quality planning "is the process necessary for identifying which quality standards are relevant to the project and determining how to satisfy them" (PMI, 2004, p. 52). Following Rose (2005, p. 42), a project management plan is created that addresses the specific project needs. Starting from a general quality policy, which outlines what quality is supposed to mean for the project, the plan defines the specific requirements and performance targets for individual parts of the project. Detailed roles and responsibilities are defined including

[1] Rose (2005) and Montgomery (2008) primarily use the term *quality control* rather than SQC. For consistency reasons, the term SQC is used throughout this thesis.

the organizational infrastructure, reporting chains and participants. A set of standards, processes and tools is identified that builds the basis for assuring the quality of the project.

Quality assurance is ”the combined set of activities that the project team will perform to meet project objectives” (Rose, 2005, p. 61). The term quality assurance is often mixed up with SQC, even though the terms identify different aspects of QM. Quality assurance basically describes the implementation of the QM plan into systematic activities and processes. For each aspect of the project, quality requirements are identified and quality assurance activities are defined in combination with appropriate quality standards, metrics and resources. (Rose, 2005, p. 61–62)

SQC addresses the outcomes of the quality assurance activities. ”It is monitoring performance and doing something about the results” (Rose, 2005, p. 61). SQC provides a set of statistical tools and methods that are being used to validate project results according to the standards and metrics that have been defined in the quality assurance plan. Corrective actions are taken if there is a gap between actual and desired quality. (Rose, 2005, p. 67)

Quality improvement is basically the application of the methods of SQC in order to achieve a continuous improvement of processes, products and services. Montgomery (2008, p. 7, 17) simply defines quality improvement as the ”reduction of variability in processes and products”, but mainly uses the term in combination with SQC by stating that together they ”involve the set of activities used to ensure that the products and services meet requirements and are improved on a continuous basis”. While SQC is a framework of statistical methods and tools, quality improvement is primarily an implementation and management strategy. Important contributors have been W. Edwards Deming with his *Deming's 14 points* (Deming, 2000) and Joseph M. Juran with his so called *Juran's trilogy* (De Feo & Juran, 2010, p. 30), who provided the ground for the *Total quality management (TQM)* strategy that began in the early 1980s, but has been only of moderate success (Montgomery, 2008).

Successful implementations of quality improvement strategies are the Toyota Production System (Ohno, 1988) and Motorola's Six Sigma (Pande et al., 2000). Interestingly, in a 2002 interview, Juran on the one hand pointed out the benefit of Six Sigma (”the concept is perfectly good”) but on the other hand expressed his doubts regarding the originality of the underlying concepts (Paton, 2002):

”*It's a basic version of quality improvement. There is nothing new there. . . One of the things that is inherent in tools used to achieve improvement under the label of Six Sigma is the concept of process capability. Now, to my knowledge, that concept of process capability goes back to 1926, when I was a young engineer at Western Electric. . . I am the inventor, if not the reinventor, of that concept.*”

So basically, the underlying statistical concepts had been available long before, but a management strategy was needed in order to apply it end to end on an organization wide basis. This is what quality improvement is about.

Among the areas of QM, the most relevant one for this thesis is SQC. The following section will illustrate the role of SQC in more detail.

4.1.2. Statistical quality control

SQC is at the heart of QM. It provides the operational tools that are identified in the quality planning phase and that are used for quality assurance and as a basis for quality improvement. Therefore, SQC represents the key concept when applying QM to cloud labor services.

The role of SQC can be illustrated based on figure 4.1 taken from Montgomery (2008, p. 13): A production process that is controlled by various input parameters is turning raw materials or components into output products. An SQC mechanism is evaluating the quality of the output products and may be taking actions on the input parameters in order to guarantee the desired level of output quality. There are two types of input parameters, those that can be controlled (*controllable inputs*) and those that cannot (*uncontrollable inputs*). Examples for controllable inputs are temperature, pressure etc., uncontrollable inputs include environmental factors and specific characteristics of the raw materials.

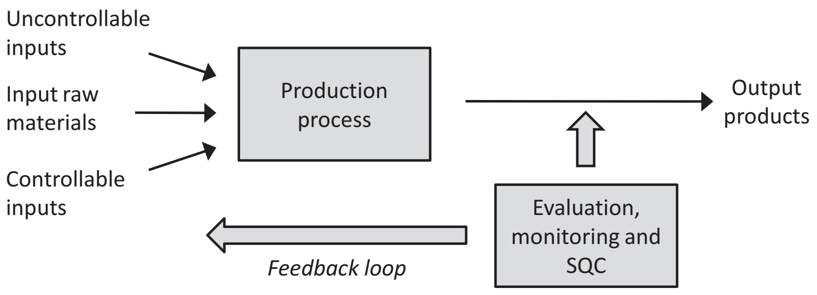


Figure 4.1.: Production process with statistical quality control; derived from Montgomery (2008, p. 13).

In any production process[2] there will always be small, unavoidable variations in the input parameters, the raw materials and the production process itself (Montgomery, 2008, p. 181). These so-called *chance causes of variation* essentially lead to an unavoidable variation of the output quality. In addition, there may be so called *assignable variations* which are being caused by avoidable variations in the input parameters, the raw materials or the production process. If a process is affected by assignable variations it is said to be *out-of-control*, if it is not, it is said to be *in statistical control*.

4.2. Areas of statistical quality control

SQC comprises three major complementary areas which are outlined in the following section: *Acceptance sampling*, *statistical process control* and *design of experiments*.

4.2.1. Acceptance sampling

Acceptance sampling is the process to decide based on a sample whether a *lot* of units meets certain quality requirements or not. If it does, the entire lot is accepted, otherwise it is rejected. Acceptance sampling does not estimate the quality of the lot or of its units but just determines whether it should be accepted or not (Montgomery, 2008, p. 632–633).

Acceptance sampling had been one of the first developments in the area of SQC. Its root goes back to Harold French Dodge's contributions in the late 1920s (Montgomery, 2008, p.

[2] The concept described in this section can be applied not only to production, but also to service processes. In that case, the raw materials may be represented by information. Section 5.1 will elaborate on that. This point of view corresponds to the modern perspective of service science, e.g. to the service dominant logic proposed by (Vargo & Lusch, 2004).

12). It is usually performed at the interface between a provider and a consumer in order to make sure that either the quality of units delivered to the consumer or the quality of units received from the provider meets certain requirements (*outgoing* vs. *incoming inspection*).

Today, acceptance sampling is mainly used in the early stages of QM efforts, when there is still limited understanding of the processes and a less intense relationship to the suppliers (Montgomery, 2008, p. 632). Over time, the SQC efforts are shifted more and more towards statistical process control and design of experiments introduced in the following. Acceptance sampling will be further elaborated on in section 4.3.

4.2.2. Statistical process control

The basic objective of statistical process control (SPC) is to recognize as quickly as possible if a process gets out of control and take corrective actions before the impact becomes too severe.

The most prominent tool for process monitoring is the *control chart*, which consists of a horizontal time axis, the so called *center line*, and two additional lines above and below the center line that represent the *upper control limit* (UCL) and the *lower control limit* (LCL). The vertical axis represents the value of a certain quality characteristic, which is determined in regular time intervals by an inspection process. The values, which are recorded on the chart over time, are naturally scattered around the center line because of chance causes of variation. The upper and the lower control limit are defined in such a way, that as long as the process is under statistical control, most of the points are plotted between them. If there are assignable root causes that cause the process to get out of control, points will be plotted outside the limits and indicate that corrective actions need to be taken. (Montgomery, 2008, p. 182–183)

SPC can be seen as the primary tool of today's quality improvement efforts (Montgomery, 2008, p. 637). Compared to acceptance sampling, the huge advantage of SPC is that it results in fewer non-conforming items at the end of the process because ”in-process inspection may reveal deficiencies that can be corrected before they cause costly scrap and rework” (Rose, 2005, p. 68).

A specific SPC tool that will be used in chapter 5 is the continuous sampling plan. It will be introduced in section 4.4.

4.2.3. Design of experiments

The idea of design of experiments is to intentionally modify the input parameters of a process in order to see what impact that will have on the output product. The insights may be used to identify patterns of root causes that affect the output quality in a specific way. The approach may also be used to develop a production process that is insensitive to certain root causes.

The primary approaches used in this thesis are acceptance sampling and continuous sampling plans which are discussed in detail in the following.

4.3. Acceptance sampling

This section introduces the area of acceptance sampling (section 4.3.1) and specifically single-sampling plans for attributes (section 4.3.2). If not indicated otherwise, the information is based on Montgomery (2008) and Rinne & Mittag (1995).

4.3.1. Introduction

The objective of acceptance sampling is to determine whether a *lot* of units is within certain quality limits without having to inspect the entire lot. Instead, only a sample of units is randomly selected from the set and is inspected according to the specified criteria. Depending on the outcome of the inspection, the entire lot is *sentenced*, i.e. it is either accepted or rejected. Compared to the 100% inspection, acceptance sampling has the advantage of reduced inspection effort and cost. A major disadvantage is the risk of accepting bad lots or rejecting good ones.

An important requirement for using acceptance sampling is that the lots need to be homogeneous, i.e. they should consist of equivalent units that have been produced by the same production process using the same machines and personnel. Processing large lots is more efficient than processing small ones because the inspection effort does not grow linearly with the lot size. In order to avoid bias, the sampling process needs to be random and representative for the entire lot.

The lot size, the sample size as well as the acceptance and rejection criteria are defined by a so-called *sampling plan*. The type of sampling plan depends on the nature of the quality characteristics that are being inspected. For continuous or countable characteristics, sampling plans for variables are used, while for a binary assessment sampling plans for attributes are used that only differentiate between good and defective units.

Another characteristic of sampling plans concerns the number of samples that are taken. When using a *single-sampling plan*, the decision between accepting and rejecting the lot is based on a single sample. With a *double-sampling plan*, the decision may already be taken based on a first sample, or depending on the outcome of the inspection, a second sample may be requested before accepting or rejecting the lot. In *multiple-sample plans*, there may even be more than two iterations. Compared to a single-inspection plan, double- or multi-sampling plans may reduce the inspection effort because the lot may be sentenced with fewer units being inspected. *Sequential-sampling plans* bring the idea to the perfection as they reduce the granularity of the samples to a minimum: They start with the inspection of an individual unit; depending on the outcome, the lot is either sentenced or an additional unit is being inspected and the process starts from scratch. This loop continues until enough information has been gathered to come to a decision. If, for example, only good or only defective items are found, there are obviously less units to be inspected than if a mix of good and defective items is found. For this thesis, the most relevant sampling plan is the single-inspection plan for attributes which is described in the following section.

4.3.2. Single-sampling plans for attributes

A single-sampling plan for attributes is a procedure where a sample of n units is drawn from a lot of size N. If the number of defects in the sample is higher than the *acceptance number* c, the lot is rejected. Otherwise, it is accepted.

Because each unit can only be drawn from the lot once, the probability for accepting or rejecting the lot can be modeled using the hypergeometric distribution. For large lot sizes N, the hypergeometric distribution can be approximated by the binomial distribution, which in turn can be approximated by a Poisson distribution.

The following sections will introduce important characteristics and types of single-sampling-plans:

Operating characteristic

The *operating characteristic* (OC) curve plots the probability L for accepting a lot depending on the actual (unknown) fraction defective p of the lot. For the binomial distribution, the graph can be calculated with

$$L(p|n;c) = \sum_{i=0}^{c} \binom{n}{i} p^i (1-p)^{n-i}\,. \tag{4.1}$$

Figure 4.2 illustrates the OC curve for different sample sizes n and acceptance numbers c. Independent of the values of n and c, the probability for accepting a perfect lot ($p = 0$) is always 1 while the probability of rejecting the worst possible lot ($p = 1$) is always 0. The higher n and the lower c, the greater is the slope between those two points and the greater is the discriminatory power. The ideal OC curve would perfectly differentiate between good and bad lots, i.e. when increasing the fraction defective p the probability of acceptance would abruptly fall from $L = 1$ to $L = 0$ at a certain $p = p_0$. The ideal OC curve can only be realized by 100% inspection.

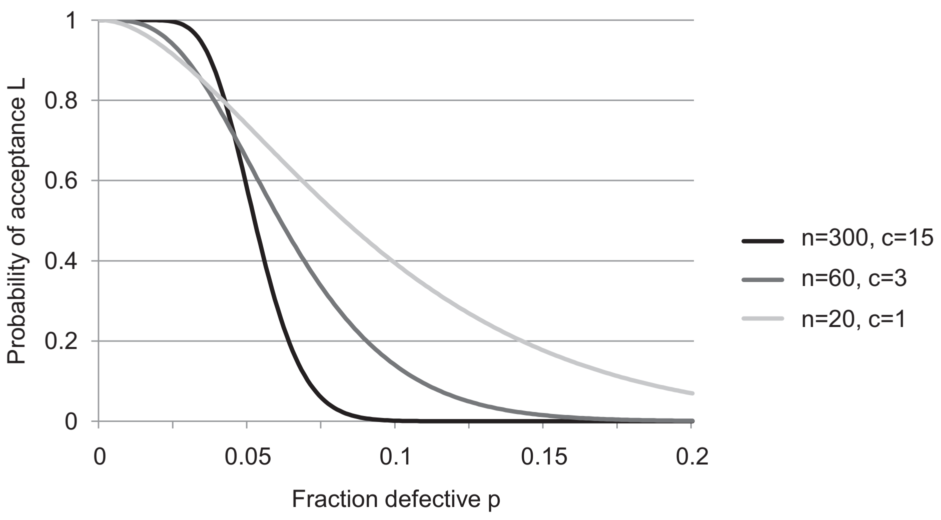


Figure 4.2.: Examples for OC curves for different combinations of sample sizes n and acceptance numbers c.

Acceptance sampling as a parameter test

Given a scenario in which the fraction defective of a lot should not exceed a critical fraction defective p_0, the decision whether to accept or to reject a lot can be seen as a parameter test on the following hypothesis:

- $H_0 : p \leq p_0$
- $H_1 : p > p_0$

According to statistical convention, the null hypothesis H_0 needs to be rejected in order to have statistical evidence for accepting the alternative hypothesis H_1. If this evidence is not found, that does not mean that there is statistical evidence for accepting the null hypothesis. The result of the parameter test does just not conflict with it.

However, even if there is statistical evidence for accepting H_1, the actual fraction defective may still exceed the critical fraction defective p_0. The two ways of arriving at wrong decisions are called type I and type II error:

- A type I error happens if hypothesis H_0 is rejected although it is actually true.
- A type II error happens if hypothesis H_0 is accepted although it is actually wrong.

Figure 4.3 illustrates the probabilities of the type I and type II errors in the above example for arbitrary fractions defective p_1 and p_2 with $p_1 \leq p_0 < p_2$. A type I error occurs if H_0 is rejected although $p_1 \leq p_0$. Its probability is α. A type II error happens if H_0 is accepted although $p_2 > p_0$. Its probability is β.

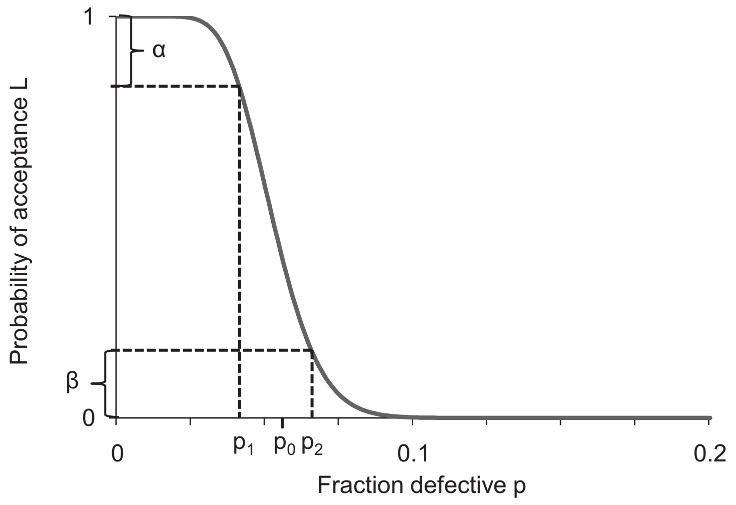


Figure 4.3.: OC curve with illustration of type I and II errors.

The acceptable and rejectable quality levels introduced in the following section are defined in a similar way.

Acceptable and rejectable quality level

The poorest quality level that the consumer would be willing to accept is called *rejectable quality level* (RQL) or *lot tolerance percent defective* (LTPD). The risk of accepting a lot with a quality level of RQL is denoted with β and is often called *consumer risk.* In order to protect the consumer, β should be low. For the producer, it is important that not too many lots are rejected. The risk of a lot being rejected is therefore often called *producer risk.* It is usually denoted with α. In order to keep it low, the quality level of the lots delivered by the producer needs to match at least the *acceptable quality level* (AQL). Figure 4.4 illustrates the points $(AQL; 1-\alpha)$ and $(RQL; \beta)$ on the OC curve.

A common approach for constructing a sampling plan is to specify the points $(AQL, 1-\alpha)$ and (RQL, β) which uniquely define the shape of the OC curve. If AQL, RQL, α and β are given, the sample size n and the acceptance number c can be calculated using the algorithm proposed by Guenther (Rinne & Mittag, 1995, p. 197), which is illustrated in form of a BPMN process in figure 4.5.

Another option is to define a limit for the average quality of the lots and to minimize the overall inspection effort, which will be described in the following paragraphs.

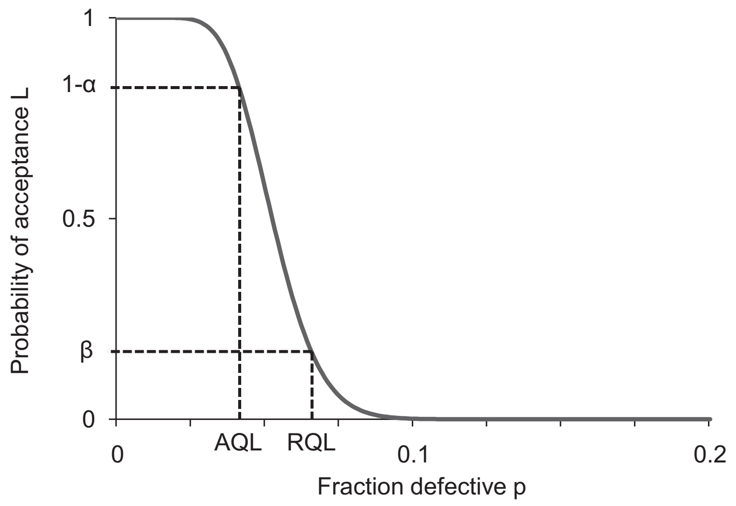


Figure 4.4.: Illustration of AQL, RQL, α and β.

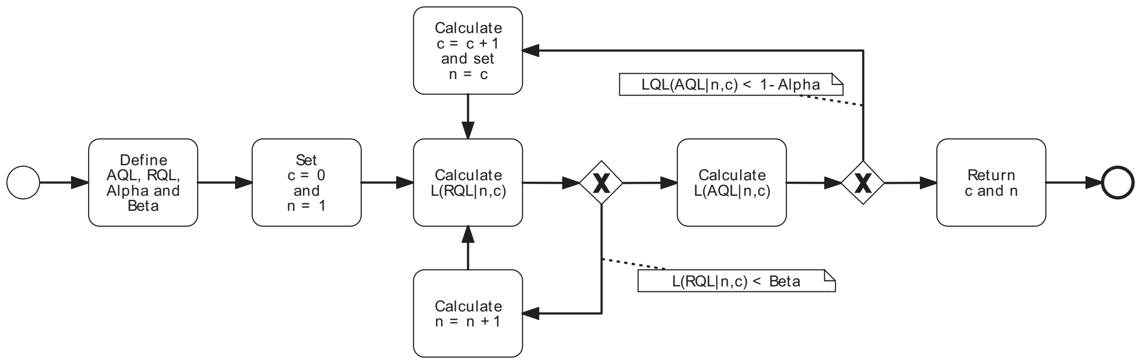


Figure 4.5.: Guenther algorithm for determining a sampling plan for given values of AQL, RQL, α and β, adopted from Rinne & Mittag (1995, p. 197).

Rectifying inspection

Two general decisions that need to be taken when applying acceptance sampling are what to do with the sample when the inspection process is completed and how to handle rejected lots. The entire sample may be discarded or only the defective units may be either removed or replaced by good ones. For rejected lots, the options are equivalent: The entire lot may be discarded or only the defective items may be removed or replaced after performing a 100% inspection of all units in the lot.

A common approach is to replace defective units in both the samples and the rejected lots. In that case, the outgoing units are a mixture of two quality levels: Rectified lots are free of defectives while accepted lots contain n good items of the corrected sample and $(N - n) \cdot p$ defective units where p is the lot's actual fraction defective. Over a longer period of time, the *average outgoing quality* (AOQ) is

$$AOQ(p) = \frac{N - n}{N} \cdot p \cdot L(p)\,. \tag{4.2}$$

Figure 4.6 illustrates the course of $AQL(p)$ versus the fraction defective p. Naturally, AQL initially increases with a growing fraction defective. However, after reaching a maximum at p_m it drops off again. This is because the worse the incoming quality of the units, the more lots are being rejected which results in a 100% inspection and a replacement of the defective units. The quality level at p_m is called *average outgoing quality limit* ($AOQL$). It is the worst average level of quality that can result from the rectifying sampling process.

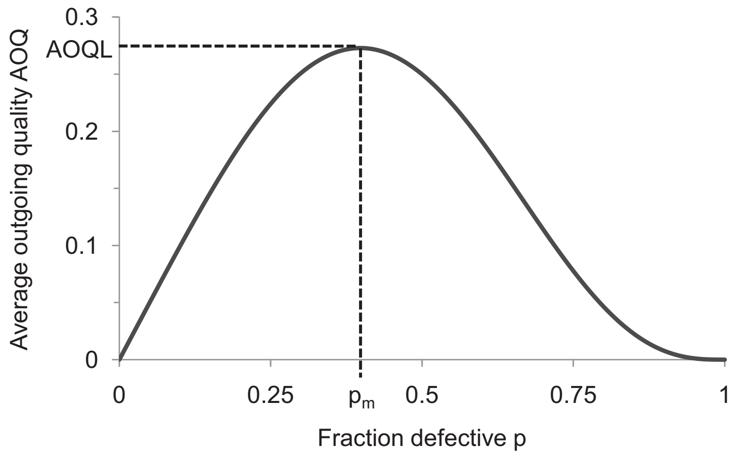


Figure 4.6.: AOQ vs. fraction defective for $n = 5$ and $c = 2$.

The overall inspection effort per lot is given by the *average total inspection* (ATI) in equation 4.3. It comprises the effort for the regular sample inspection, for the 100% inspection of rejected lots and for the screening of units that is needed to replace defective units by good ones. For a perfect lot, it equals the sample size n. If the fraction defective is close to 1, ATI dramatically increases, mainly because of the screening effort.

$$ATI(p) = \frac{n + (N - n) \cdot (1 - L(p))}{1 - p} \tag{4.3}$$

The *average fraction inspected* (AFI) is defined as the inspection effort per unit:

$$AFI(p) = \frac{ATI(p)}{N} \tag{4.4}$$

The Dodge-Romig AOQL sampling plan

Defining $AOQL$ is not sufficient for constructing a sampling plan, but an additional constraint is required. A common approach is to choose the sampling plan in a way that the inspection effort is minimized by minimizing ATI. The Dodge-Romig AOQL plan (Dodge & Romig, 1959) addresses that objective. It relies on an estimate for the *process average p* which is the expected fraction defective of the incoming units. The parameters of the sampling plan cannot be calculated by a simple formula but can be looked up in Dodge & Romig (1959) or Montgomery (2008). Examples for an $AOQL = 10.0\%$ and a process average of 8.01% to 10.00% can be found in table 4.1. The table provides the sample size n and the acceptance number c for different lot sizes N. If the process average is estimated incorrectly, the quality objective can still be assumed to be met but the process may result in a higher ATI.

For instance, for a lot size of $N = 40$, a sample of $n = 7$ items would be drawn. If more than $c = 1$ items are found defective, the entire lot is rejected. Otherwise it is accepted. This process would guarantee an $AOQL$ of 10% at most.

Table 4.1.: Dodge-Romig example inspection plans for an *AOQL* of 10.0% and a process average of $8.01\% \leq p \leq 10.00\%$ at different ranges of the lot size N; excerpt of Dodge & Romig (1959, p. 204).

N_{min}	N_{max}	n	c
1	3	*all*	0
4	50	7	1
51	100	12	2
101	200	18	3
201	300	23	4
301	400	29	5
401	500	30	5

4.4. Continuous sampling plans

Acceptance sampling plans are subject to an important restriction: If the units do not occur in batches, but in a continuous production process, such as in line assembly or in a service scenario, the process has to be decomposed into artificial batches. Before a whole batch has been handled, quality levels for this batch cannot be guaranteed and the results of this batch cannot be further processed. However, the batches cannot be arbitrarily small, because the smaller the batches, the less efficient is the sampling process.

The concept of a *continuous sampling plan* (CSP) has been developed in order to overcome the restriction that acceptance sampling cannot be applied to continuous production processes. CSPs can guarantee the average quality of a continuous stream of items at any point in time. As the items are coming in, a simple process decides for each individual item, whether it has to be inspected or not in order to guarantee the average quality of the entire stream.

4.4.1. The continuous sampling plan 1 (CSP-1)

The CSP-1 (Dodge, 1943) is the very first version of a CSP that has been developed. Although it was later on refined by Dodge himself and others (Dodge & Torrey, 1951; Lieberman & Solomon, 1955; Gosh, 1996), the CSP-1 can be assumed to be still the most widespread implementation of a CSP.

The CSP-1 is designed for attributes, i.e. each individual item is categorized as either good or defective. The defect probability p is assumed to follow a Bernoulli distribution and the process of incoming items is assumed to be under statistical control, which means that the incoming fraction defective p does not change over time. Furthermore, it is assumed that the sample inspection is perfect and defective units are replaced by good ones.

As illustrated by figure 4.7, the sampling plan starts with 100% inspection. If i consecutive units have been identified to be free of defects, the process turns into fractional inspection mode with an inspection probability f, i.e. only a portion f of the items is inspected. Once, a defective item is found during the fractional inspection mode, the process switches back to 100% inspection mode and the process starts from scratch. The parameter i is called the *clearance number* and f is called the *sampling fraction.* The most important characteristic of the CSP-1 is the *average outgoing quality limit* (*AOQL*). It is the highest average amount of defective units capable of passing through without being inspected that can be reached depending on the incoming fraction defective p.

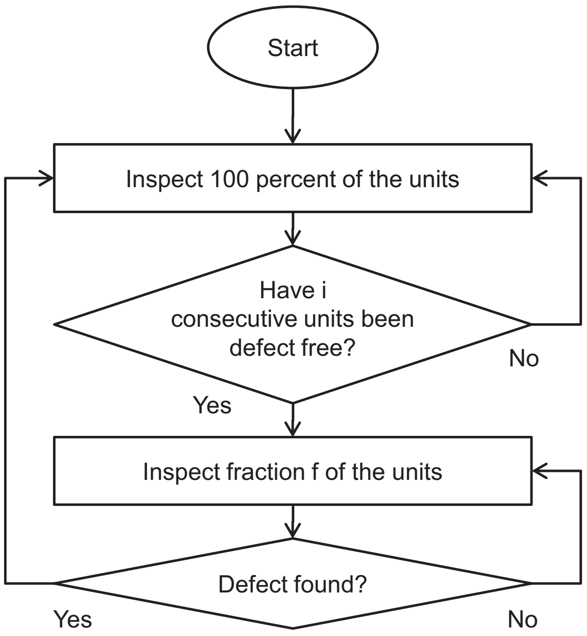

Start
Inspect 100 percent of the units
Have i consecutive units been defect free?
No
Yes
Inspect fraction f of the units
Defect found?
Yes
No

Figure 4.7.: Procedure of the CSP-1, taken from Montgomery (2008, p. 684).

There are multiple combinations of the parameters i and f which result in the same value of $AOQL$. For a given $AOQL$, one of the parameters i and f can be chosen depending on the needs of the scenario and the corresponding parameter can then be calculated according to (4.5):

$$f = \left(1 + \frac{\left(1+\frac{1}{i}\right)^{i+1} \cdot i \cdot AOQL}{(1-AOQL)^{i+1}}\right)^{-1} \tag{4.5}$$

4.4.2. Determination of clearance number and sampling fraction

This section describes two ways of choosing the clearance number i and the sampling fraction f which will be needed for the model in section 5.

Imperfect sample inspection

In many scenarios, the inspection of the samples is not perfect, but an *inspection error* applies to the sampling process. In case of imperfect inspection, two types of inspection errors can be made:

- E_1: a good item is classified as defective (type 1 inspection error).
- E_2: a defective item is classified as good (type 2 inspection error).

According to Wang & Chen (1997), under the assumption of imperfect inspection, (4.5) is replaced by (4.6):

$$f = \frac{(1-(1-e_2)\hat{p}-(1-\hat{p})e_1)^i \cdot (1-\frac{AOQL}{\hat{p}})}{((1-(1-e_2)\hat{p}-(1-\hat{p})e_1)^i - 1) \cdot (1-\frac{AOQL}{\hat{p}}) + (1-e_2)} \tag{4.6}$$

The parameters $e_1 = P(E_1)$ and $e_2 = P(E_2)$ are the probabilities of a type 1 and type 2 inspection error. $AOQL$ is the average outgoing quality limit and $\hat{p}$ is the expected average incoming fraction defective, i.e. the probability of processing a defective item.

Short sequences of low quality

One possible objective of choosing i is to make the CSP-1 resistant to short sequences of low quality (Rinne & Mittag, 1995): Let $P^* \in \,]0,1]$ be the (low) quality level of a (short) series of l items within the continuous stream of items. In order to ensure that the probability of accepting such a series is not higher than $\tilde{w}$, the sampling fraction f must be

$$f = \frac{1}{P^*}\left(1 - \tilde{w}^{\frac{1}{l}}\right) \, . \tag{4.7}$$

The corresponding clearance number i can be identified with (4.5) or (4.6) by inserting different values of i until an f close to the desired one has been found. As i must be an integer, only discrete values of f are possible.

Part II.

Model

5. Core model for statistical quality control of cloud labor services

This chapter[1] describes a new model for crowd-based QM for cloud labor services that represents the primary contribution of this thesis. The objective of the model is to overcome the restrictions of the existing QM approaches for cloud labor services described in chapter 3 by guaranteeing a certain well-defined long-run average quality for a continuous stream of results, while minimizing the costs in terms of labor work. An additional objective of the model is to provide means for influencing the completion time for a given number of tasks. The new model combines statistical quality control with a newly developed dynamic voting mechanism (DVM).

After motivating the approach in section 5.1, the actual model is introduced in section 5.2. Section 5.3 then specifically focuses on the DVM while section 5.4 addresses the objective of influencing the completion time. Finally, section 5.5 explains how the approach can be applied to specific scenarios and section 5.6 provides additional considerations for operating the model with a small number of workers.

5.1. Motivation

The analysis in section 3.4.2 has revealed that there is a lack of efficient, scalable and goal-based QM mechanisms for cloud labor services. SQC represents a promising methodology for closing this gap because, according to chapter 4.1.2, efficiency and the ability of supporting well-defined quality objectives are key characteristics of SQC. Beyond that, the general setup is similar: Like in production processes for which SQC was originally designed, there is typically a large number of similar items (results) being produced by cloud labor services. The model introduced in this chapter applies the concept of SQC to cloud labor services in order to guarantee a well-defined result quality at minimum worker effort.

Section 5.1.1 illustrates, how SQC can be adopted to cloud labor services while section 5.1.2 specifically addresses the challenge of sample inspection in cloud labor scenarios.

[1] Considerable parts of this chapter have already been exposed to and tested with the academic community, having been published in the International Journal of Cooperative Information Systems (Kern et al., 2012b).

5.1.1. Application of statistical quality control

As a starting point for applying quality control to cloud labor services, the generic production process illustrated by figure 4.1 in section 4.1.2 is used. In the cloud labor scenario, the production can be represented by a worker who performs tasks on a cloud labor platform and returns responses. The characteristics of the worker, his availability as well as the variability and individual difficulty of the tasks can be regarded as uncontrollable inputs. The task design and instructions, the payment as well as the admission of a worker can be seen as controllable inputs. The raw materials are represented by the task data, which is the portion of information that differs from task to task in a stream of similar tasks. In an image recognition scenario it could be the actual image to be processed. Finally, the quality control component inspects the worker's responses and ensures that only such responses are delivered as results that adhere to certain quality objectives. The results can be regarded as the output products of the process. Figure 5.1 illustrates the scenario.

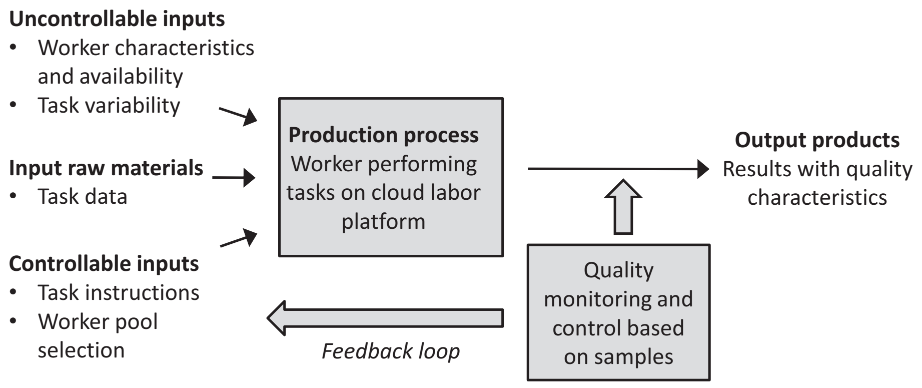


Figure 5.1.: Production process of SQC applied to the cloud labor scenario.

In the *optical character recognition* (OCR) scenario that is to be used in the evaluation in chapter 8, the raw materials are pictures of handwritten words which are passed to workers along with the instruction to read them and type them into an input field presented on the computer screen. The output products are the digital representations of the words returned by the workers.

So far, applying the concept of quality control to cloud labor services appears to be straight forward. However, a remaining challenge concerns the inspection of the worker results. In the OCR scenario, the requester would usually not know the digital representation of the handwritten word, otherwise there would be no need to apply crowdsourcing. For the same reason, it is usually not possible to assess the accuracy of a response in an automated way (Hirth et al., 2013) and a manual inspection by the requester would not scale either.

The approach taken in this thesis is to perform the sample inspection by applying one of the QM patterns for cloud labor services introduced in section 3.3. In order to explore different combinations, two SQC mechanisms are investigated along with two different QM patterns. They are depicted by table 5.1. Two basic SQC mechanisms have been chosen out of the primary areas of SQC: The CSP-1 out of the area of *statistical process control*, and the *Dodge-Romig AOQL sampling plan* out of the area of *acceptance sampling*. The QM patterns to be investigated are *voting* and *validation*. In this chapter, the combination

Table 5.1.: Combinations of SQC mechanisms and QM patterns for cloud labor services to be investigated in this thesis.

		Quality management patterns (chapter 3)	
		Voting pattern	Validation pattern
SQC mechanism (chapter 4)	Statistical process control	CSP/DVM (chapter 5)	
	Acceptance sampling		Group validation approach (section 6.2)

of the CSP-1 and the voting pattern will be investigated while the combination of the Dodge-Romig AOQL sampling plan and the validation pattern will be investigated when developing the *group validation approach* in section 6.2. The rationale for choosing the SQC mechanisms will be discussed when introducing the respective models. The choice of the QM patterns will be motivated in the following section.

5.1.2. Quality management patterns for sample inspection

According to the task characteristics defined in section 3.3.1, sample inspection can be seen as a *simple*, *deterministic* and *fine-grained* task. Following the decision matrix in table 3.3 on page 60, the two recommended QM patterns for this type of tasks are *validation* and *voting*. Those are the two patterns to be applied in this thesis. This decision will be further motivated by looking at each of the five available patterns:

The *iteration* and the *comparison* patterns are obviously not suitable because their actual strength is to produce responses rather than to validate them. For QM purposes both patterns internally rely on a voting step.

The *gold* pattern appears to meet the requirements of the sample inspection process well. By systematically interspersing gold standard tasks into the stream of tasks, samples could be inspected by comparing the appropriate worker responses with the ground truth of those tasks. However, according to section 3.3.2, the gold standard tasks would need to be expensively created with reasonable manual effort. Therefore, the gold pattern would limit the scalability of the QM mechanism.

The *validation* pattern probably represents the most obvious match because validation is close to the idea of inspection. However, a single worker out of the crowd cannot be assumed to meet the requirements of the inspection process with regard to the accuracy of the validation decision. Therefore, the pattern will be used along with the voting pattern when developing the *group validation approach* in chapter 6.2.

For the *voting* pattern, it may not be obvious how it can be applied to SQC at all. According to figure 5.1, each worker represents a separate production process for which a sample inspection mechanism needs to be established. The basic approach being used here is exploiting the fact that with the voting pattern, a consolidated reliable response can be generated from the contributions of multiple workers which in turn can be used to inspect these individual contributions. While SQC is used on a per worker basis, the voting mechanism is used across a set of workers. Figure 5.2 illustrates this principle:

Assignments of the same task are passed to multiple workers in step 1 and their responses are aggregated into a single consolidated response in step 2. Finally, the consolidated response is used as a reference for inspecting the individual worker contributions in step 3.

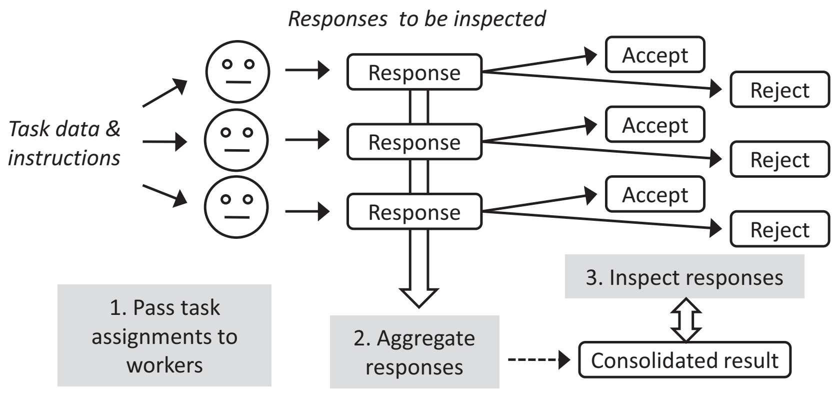


Figure 5.2.: Basic schema of task inspection based on the voting pattern.

5.2. Model

This section describes the construction of the core QM model of this thesis as a combination of the continuous sampling plan CSP-1 and the dynamic voting mechanism (DVM).

After introducing the underlying assumptions in section 5.2.1, section 5.2.2 illustrates the process flow of the model. Section 5.2.3 then describes the rationale for choosing the CSP-1 as the SQC component to be used in the model.

5.2.1. Assumptions

Referring to section 2.1.2, the basic scenario of cloud labor services comprises three roles: the service *requester*, the *cloud labor platform* and the *workers*. The platform acts as an intermediary between the requester who publishes tasks and workers who pick tasks and work on them in return for a compensation per task. There is assumed to be a large number of equivalent tasks of the same *task type* that consist of the same *task instructions* but different *task data*. The model makes some additional assumptions about the underlying platform, the tasks as well as the workers which are described in the following.

Platform

In accordance with existing cloud labor platforms like MTurk it is assumed that the platform allows for tracking individual workers based on an individual *worker ID* which is returned to the requester for each response delivered by the worker. The platform also supports multiple redundant *assignments* of the same task and ensures that they are completed by different workers in order to use the responses for voting steps. There are also means for defining worker pools i.e. for making specific tasks only available to a subset of the workers. MTurk implements this by offering qualification tests which also ensure that the workers at least initially fall below a certain failure rate when working on a specific type of tasks.

Tasks

Because the QM approach presented here is a form of the voting pattern, it is restricted to deterministic[2] tasks. Therefore, redundant responses delivered by multiple workers for assignments of the same task can be compared to each other or be aggregated into a single consolidated result. It is further assumed that a large number of similar tasks are available that require similar skills and that a fixed compensation is paid to the workers per task independent of the quality of the work results.

Workers

From a statistical perspective it is important that the workers are working independently of each other. Most importantly, a worker who is working on a specific task must not collaborate (e.g. agree on a response) with other workers working on assignments of the same task. Therefore, there should be a large crowd of workers who do not know each other. Another assumption is that a failure rate can be attached to each worker which is the same for all tasks of a given task type and which only slowly changes over time, for example because workers are improving their skills or because they are getting tired after performing a large number of tasks.

5.2.2. Process flow

The CSP/DVM can be seen as a QM component on top of the basic cloud labor platform outlined in section 5.2.1. It consists of two functional modules: A state-of-the-art quality control mechanism and a newly developed dynamic voting mechanism (DVM). The quality control module uses the CSP-1 which has been described in section 4.4. A separate CSP-1 is used for each worker. The dynamic voting module will be described in section 5.3. The overall scenario is illustrated by figure 5.3: A requester submits a task to the QM component which immediately publishes a first assignment of it to the cloud labor platform. A worker grabs the task assignment, works on it and returns a response which is fed back to the QM component in combination with the worker's ID. Depending on the CSP status of the worker a sample inspection is being initiated or not. If not, the worker's response is accepted as the final result and is passed back to the requester without further validation. If an inspection is required, the DVM is initiated.

The DVM publishes another assignment of the same task and an additional response is returned by a different worker via the cloud labor platform. Based on the two available responses and the historical failure rates of the two workers, the response with the highest probability of correctness is identified. If this probability equals or exceeds the predefined minimum inspection quality, the corresponding response is accepted as the final result and returned to the requester. In addition, the failure rates and the CSP status of the participating workers are updated depending on whether they had returned the correct response or not. If the predefined minimum inspection quality has not been reached yet, the redundancy is increased by publishing another assignment of the same task, receiving a response from another worker and again identifying the response with the highest probability of correctness. This process is continued until *either* the minimum inspection quality is met *or* it becomes too unlikely for the involved workers to come only to the observed level of agreement given their historical failure rates. In the latter case, it can be assumed that there is something wrong with the task or with the task instructions. Therefore, the task is escalated back to the requester so he can validate it or improve the instructions.

[2] Refer to section 3.3.1.

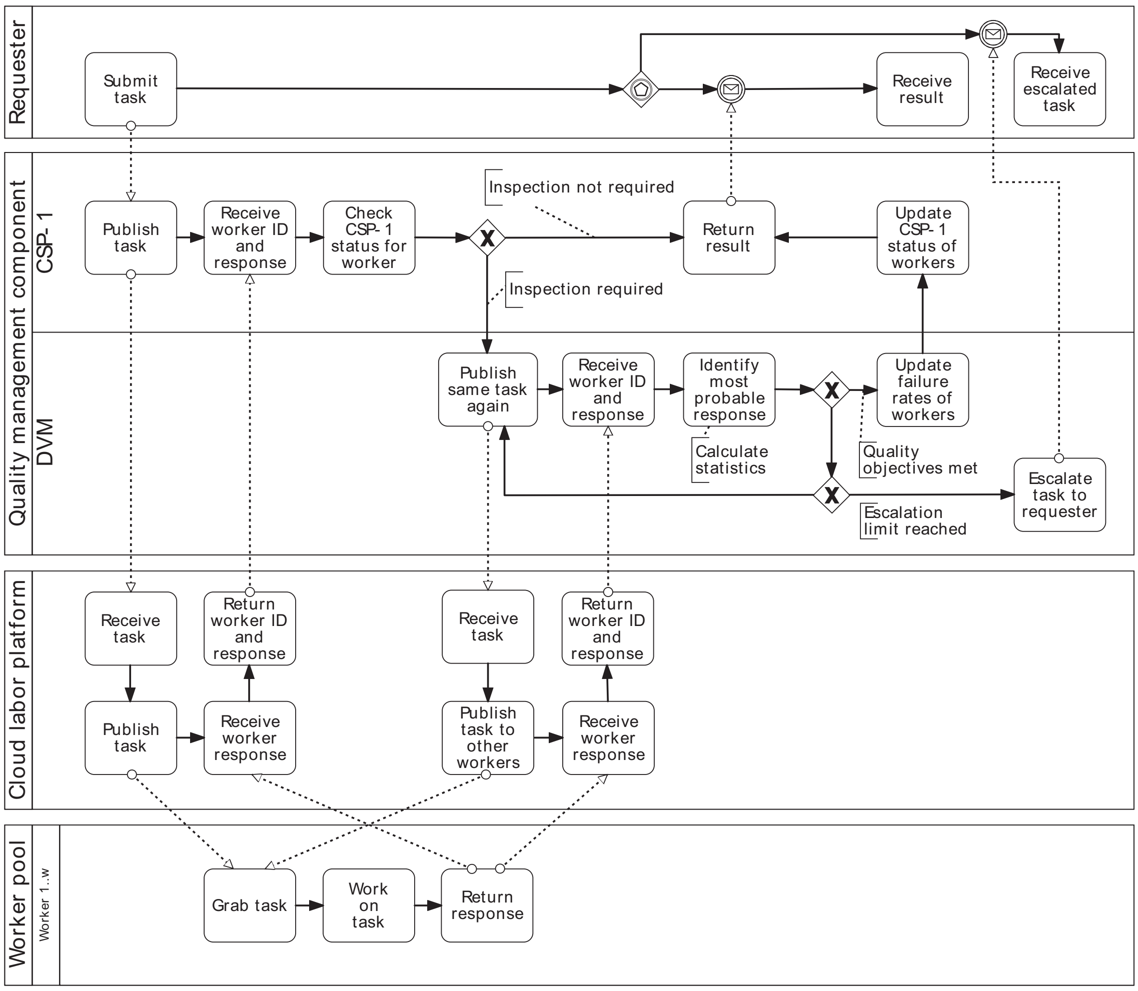


Figure 5.3.: Process flow of the CSP/DVM in the context of the service requester, the cloud labor platform and the crowd workers.

Figure 5.4 illustrates the matrix of worker contributions per task. The horizontal axis represents the id of the tasks as they are picked up by the workers while the vertical axis represents the workers in the order they have joined the process. Each circle represents a single task assignment. A full circle indicates the initial assignment of a task while empty circles indicate redundant assignments issued by the DVM. In the example, only a single worker is contributing to tasks 1, 4, 5, 7, 10 and 11 respectively, which indicates that the CSP-1 of the corresponding workers did not trigger an inspection. Apparently, the CSP-1 of those workers is in *fractional inspection* mode. For tasks 2, 3, 6, 8 and 9, an inspection was triggered by the CSP-1 of the workers who processed the initial assignment. Therefore, the DVM was issued and responses for additional assignments were requested (see tasks 3, 6 and 9). For instance, three additional workers contributed to task 9 after worker 3 had returned the initial assignment. It seems that the CSP-1 of worker 3 is in full inspection mode because all of his initial assignments are inspected.

It is essential to understand that even though they are presented in one column, the assignments of a given task are not performed simultaneously but sequentially. Figure 5.5 illustrates the processing timeline of the example in figure 5.4. The vertical axis again represents the worker ids whereas the horizontal axis now represents time. The width

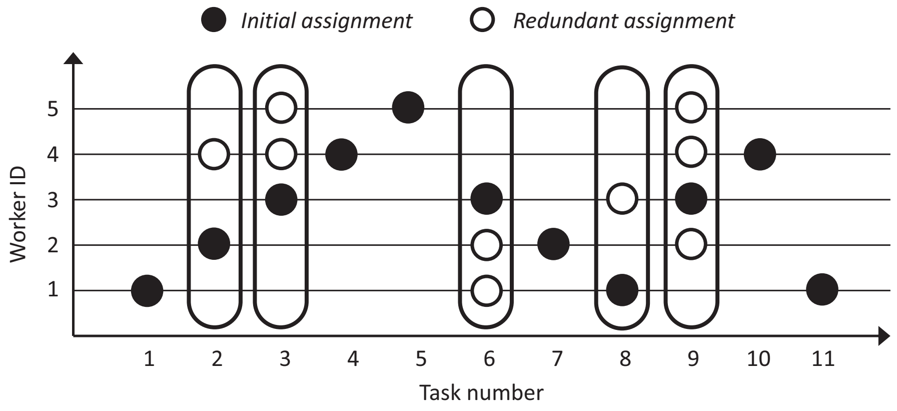


Figure 5.4.: Illustration of worker contributions to a set of tasks.

of the rectangles illustrates the duration a worker has worked on the assignment of a given task. Full rectangles indicate initial assignments while empty rectangles indicate redundant assignments. Workers 1 and 2 seem to be rather slow while worker 3 seems to be extraordinarily fast. According to the process flow in figure 5.3, redundant assignments are only triggered after previous assignments of a task have been completed. A DVM is started every time a redundant assignment is completed by a worker, which is indicated by a thin vertical line in the timeline. A result is returned to the requester whenever an assignment is finished and no additional redundancy is needed (indicated by an asterisk in the task bar). This happens when either the CSP-1 does not trigger an inspection of the corresponding worker or when the DVM decides that no more assignments are needed.

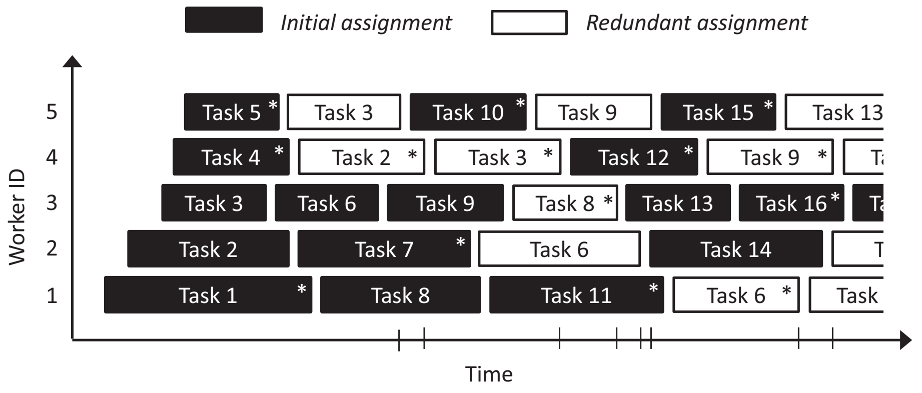


Figure 5.5.: Processing timeline of the example in figure 5.4.

Before section 5.5.3 will explain the process flow in more detail, the following section will motivate the use of the CSP-1.

5.2.3. Rationale for using the CSP-1

The CSP-1 was chosen for a number of reasons: First, it is not limited to batch scenarios but supports real-time cloud labor scenarios in which a continuous stream of tasks needs to be completed and for which the response time matters. Second, it works well with the DVM: The CSP-1 assumes that defective items are replaced. So if an incorrect worker response is detected, it will be replaced by the correct one. The DVM supports that requirement as it implicitly identifies the (most probable) correct result during the sample inspection process. Finally, extensions for imperfect sample inspection are available for the CSP-1, which is an important requirement of the DVM as it only supports a limited inspection quality level.

The assumption of the CSP-1 that the quality of the incoming worker results is under statistical control is addressed in three ways: First, it is assumed that all tasks are similar and require similar skills. Second, an escalation mechanism is provided that identifies outliers. Finally, if sequences of exceptionally bad worker results are expected, the CSP-1 parameters may be specified in a way that protects it against them.

5.3. Dynamic voting mechanism (DVM)

This section provides the statistical foundation for the DVM. After defining a set of terms in section 5.3.1, the model is developed in section 5.3.2 and the results are presented in section 5.3.3.

5.3.1. Definitions

Let $A = \{a_1, a_2, .., a_v\}$ be the set of all v possible responses to a task. According to the assumptions described in section 5.2.1, exactly one of the possible responses $a \in A$ is the expected correct response for the task. Let

$$\hat{R} = A^w \tag{5.1}$$

be the set of all possible response tuples that might be returned by a group of w different workers who have worked on assignments of the same task. Let $R \in \hat{R}$ be a concrete response tuple with $R = (r_1, r_2, .., r_w)$ and let the set $D = \{r_1, r_2, .., r_w\}$ consist of the distinct responses in R. Furthermore, let the tuple $E = (p_1, p_2, .., p_w) \in \,]0; 1[^w$ represent the individual historical failure rates $p_1, p_2, .., p_w$ of the contributing workers.

Let $\hat{C} = \{0, 1\}^w$ and $C \in \hat{C}$ be the *correctness profile* of the task. The tuple $C = (c_1, c_2, .., c_w)$ indicates, which worker has returned a correct versus an incorrect response, i.e.

$$\forall_{j=1..w} : c_j = \begin{cases} 0 & \text{if worker } j \text{ has returned an } \textit{incorrect} \text{ response} \\ 1 & \text{if worker } j \text{ has returned the } \textit{correct} \text{ response}. \end{cases} \tag{5.2}$$

As an example, the correctness profile $(1, 0)$ represents the case that the first of two workers returns a correct response, while the second worker returns an incorrect response.

5.3.2. Statistical considerations

The set $S = \hat{R} \times \hat{C}$ spans the sample space of all possible response tuples $R \in \hat{R}$ and correctness profiles $C \in \hat{C}$. An exemplary sample space for 2 workers is provided by table 5.2, an example for 3 workers by table A.1 in appendix A. The joint probabilities $P_E(C \cap R)$ in each row sum up to the overall probabilities $P_E(R)$ of the response tuples R and the joint probabilities in each column sum up to the overall probabilities $P_E(C)$ of the correctness profiles. The index E indicates that all probabilities are conditional given the failure rates E of the contributing workers. The following paragraphs describe how the individual probabilities can be calculated.

Depending on their failure rates, an arbitrary subset of the workers might return a correct response. As the probability for worker j to deliver a correct vs. incorrect response is

Table 5.2.: Sample space of a scenario with 2 workers and 3 possible responses assuming a historical failure rate of $p = 0.1$ for both workers.

	$P_E(C \cap R)$ for correctness profile C				
Response tuple R	$(0,0)$	$(0,1)$	$(1,0)$	$(1,1)$	$P_E(R)$
(a_1, a_1)	0.0011	0	0	0.2700	0.2711
(a_1, a_2)	0.0011	0.0150	0.0150	0	0.0311
(a_1, a_3)	0.0011	0.0150	0.0150	0	0.0311
(a_2, a_1)	0.0011	0.0150	0.0150	0	0.0311
(a_2, a_2)	0.0011	0	0	0.2700	0.2711
(a_2, a_3)	0.0011	0.0150	0.0150	0	0.0311
(a_3, a_1)	0.0011	0.0150	0.0150	0	0.0311
(a_3, a_2)	0.0011	0.0150	0.0150	0	0.0311
(a_3, a_3)	0.0011	0	0	0.2700	0.2711
$P_E(C)$	0.0100	0.0900	0.0900	0.8100	1.0000

$q_j = (1 - p_j)$ vs. p_j, the a priori probability $P_E(C)$ for observing a correctness profile C can be estimated as

$$P_E(C) = \prod_{\forall j | c_j = 1} q_j \prod_{\forall j | c_j = 0} p_j \,. \tag{5.3}$$

In order to define the most probable response, the conditional probabilities $P_E(C \mid R)$ of all possible correctness profiles given a specific response tuple R and the tuple of failure rates E are calculated and then the correctness profile $\tilde{C}$ with the highest probability is identified. The estimation is being performed with Bayesian inference using the a priori information about the failure rates of the involved workers and about the distribution of the possible response tuples. $P_E(C \mid R)$ can be calculated as the conditional probability for C given R and E:

$$P_E(C \mid R) = \frac{P_E(C \cap R)}{P_E(R)} = \frac{P_E(R \mid C) P_E(C)}{P_E(R)} \tag{5.4}$$

Within (5.4), the conditional probability $P_E(R \mid C)$ can be calculated as

$$P_E(R \mid C) = \begin{cases} \frac{1}{m_C} & \text{if } \exists k \in D \mid \forall j : (c_j = 1) \Rightarrow (r_j = k), (c_j = 0) \Rightarrow (r_j \neq k) \\ 0 & \text{otherwise} \end{cases} \tag{5.5}$$

with m_C being the number of response tuples R that may be observed given a specific correctness profile C. The underlying assumption is that the a priori probability of all response tuples is equal. For example in table 5.2, if there are three possible responses a_1, a_2 and a_3, there are three response tuples in $\hat{R}$, that apply to the correctness profile $(1,1)$: (a_1, a_1), (a_2, a_2) and (a_3, a_3). Other response tuples like (a_2, a_3) have a probability of zero because a_2 and a_3 cannot be correct at the same time as there is only a single correct response $a \in A$.

The number m_C is determined by

$$m_C = \begin{cases} v \cdot \prod\limits_{\forall l | c_l = 0} (v-1) & \text{if } \exists j \mid c_j \neq 0 \\ v^w & \text{if } (\forall j \mid c_j = 0) \wedge (v > w) \\ v^w - \sum\limits_{l=0}^{v} \binom{v}{l} (v-l)^w (-1)^l & \text{if } (\forall j \mid c_j = 0) \wedge (v \leq w) \,. \end{cases} \tag{5.6}$$

In the upper branch of (5.6), the response tuple contains the (unknown) correct response $a \in A$ at least once, so there are v options to choose it and $(v-1)$ options for choosing each of the incorrect responses. In the middle and lower branch, all responses of the response tuple are incorrect. If there are more possible responses v than workers w, any of the v^w response tuples in $\hat{R}$ might be incorrect. If the number of possible responses is lower than or equal to the number of workers, *Stirling numbers of the second kind* (Graham et al., 1994) are used to exclude those response tuples that contain all possible responses out of A. They cannot occur with the correctness profile $C = (0, 0, .., 0)$ because one of the responses $a \in A$ must be correct.

The marginal probabilities $P_E(R)$ are determined by

$$P_E(R) = \sum_{\forall C \in \hat{C}} P_E(R \mid C) P_E(C) \,. \tag{5.7}$$

5.3.3. Calculation of the conditional probabilities

Depending on the number of possible responses v and the number of workers w, the matrix in table 5.2 can become rather big. However, in order to determine the conditional probability $P_E(C \mid R)$, it is not required to calculate the entire matrix because according to (5.5), $P_E(R \mid C)$ and therefore $P_E(C \mid R)$ is different from zero only for such $C_{k,R} \in \hat{C}$ for which all correct responses can be mapped to a specific distinct response $k \in D$ and none of the incorrect responses is equal to k. Therefore, (5.4) can be written as:

$$\varphi_k := P_E(C_{k,R} \mid R) = \frac{P_E(C_{k,R})}{m_{C_{k,R}} \cdot P_E(R)} \tag{5.8}$$

with

$$P_E(C_{k,R}) = \prod_{\forall j | r_j = k} q_j \prod_{\forall j | r_j \neq k} p_j \,. \tag{5.9}$$

Using (5.5) through (5.9), this turns into the following two equations:

1. If there are more possible responses than workers, i.e. $v > w$, that results in

$$\varphi_k = \frac{\prod\limits_{\forall j | r_j = k} q_j \prod\limits_{\forall j | r_j \neq k} p_j}{v \cdot \prod\limits_{\forall j | r_j \neq k} (v-1) \cdot \left(\sum\limits_{k \in D} \frac{\prod_{\forall j | r_j = k} q_j \prod_{\forall j | r_j \neq k} p_j}{v \cdot \prod_{\forall j | r_j \neq k} (v-1)} + \frac{\prod_{\forall j} p_j}{v^w} \right)} \,. \tag{5.10}$$

2. If the number of possible responses is less than or equal to the number of workers, i.e. $v \leq w$, it turns into

$$\varphi_k = \frac{\prod\limits_{\forall j | r_j = k} q_j \prod\limits_{\forall j | r_j \neq k} p_j}{v \cdot \prod\limits_{\forall j | r_j \neq k} (v-1) \cdot \left(\sum\limits_{k \in D} \frac{\prod_{\forall j | r_j = k} q_j \prod_{\forall j | r_j \neq k} p_j}{v \cdot \prod_{\forall j | r_j \neq k} (v-1)} + \frac{\prod_{\forall j} p_j}{v^w - \sum_{l=0}^{v} \binom{v}{l} (v-l)^w (-1)^l} \right)} \,. \tag{5.11}$$

An example calculation of $\varphi_k = P_E(C_{k,R} \mid R)$ for the sample space of 2 workers is provided in table 5.3 and for the sample space of 3 workers in table A.2 in appendix A. Table 5.3 can be derived from table 5.2 according to (5.4) by dividing the values $P_E(C \cap R)$ in each table cell by the corresponding value $P_E(R)$ in the last column. For example, for the response tuple $R = (a_1, a_1)$ and the correctness profile $C = (1, 1)$, the conditional probability is

$$P_E(C \mid R) = \frac{P_E(C \cap R)}{P_E(R)} = \frac{0.2700}{0.2711} = 0.9959\,. \tag{5.12}$$

In the scenario of two workers with a failure rate of $p = 0.1$ each and three possible response options, the number 0.9959 represents the probability that both workers return the correct response. The three response options could be answers A, B and C in a multiple choice test where only one of them can be correct.

Table 5.3.: Conditional probability $P_E(C \mid R)$ for observing a certain correctness profile C given a specific response tuple R in a scenario with 2 workers and 3 possible responses assuming a historical failure rate of $p = 0.1$ for both workers.

	$P_E(C \mid R)$ for correctness profile C			
Response tuple R	$(0,0)$	$(0,1)$	$(1,0)$	$(1,1)$
(a_1, a_1)	0.0041	0	0	0.9959
(a_1, a_2)	0.0357	0.4821	0.4821	0
(a_1, a_3)	0.0357	0.4821	0.4821	0
(a_2, a_1)	0.0357	0.4821	0.4821	0
(a_2, a_2)	0.0041	0	0	0.9959
(a_2, a_3)	0.0357	0.4821	0.4821	0
(a_3, a_1)	0.0357	0.4821	0.4821	0
(a_3, a_2)	0.0357	0.4821	0.4821	0
(a_3, a_3)	0.0041	0	0	0.9959

The response $d \in D$ with the highest probability of correctness can be calculated as

$$d \in D \mid \forall k \in D : \varphi_d \geq \varphi_k\,. \tag{5.13}$$

If the probability φ_d equals or exceeds φ_{min}, the response d is accepted as the correct response. The escalation limit is defined as the probability for observing the correctness profile $C_{d,R}$ given the failure rates E of the involved workers

$$\varepsilon_d := P_E(C_{d,R}) = \prod_{\forall j \mid r_j = d} q_j \prod_{\forall j \mid r_j \neq d} p_j\,. \tag{5.14}$$

Once the probability ε_d underruns ε_{min}, the task is escalated back to the requester. The rationale of this definition of the escalation limit is the following: In general, the QM mechanism relies on the assumption that the historical failure rates of the workers are good estimates for their future performance. However, there may be some tasks which require higher or different skills or which are not solvable at all. For those tasks, the historical worker failure rates are no good estimates and, therefore, the probability $P_E(C_{d,R})$ of the correctness profile can be much lower than for typical tasks.

5.4. Completion time management

This section provides considerations about influencing the completion time of a given batch of tasks by controlling the number of workers who are eligible to work on the tasks. It is shown that applying a deadline can further increase the efficiency of the model.

After motivating the considerations in section 5.4.1, sections 5.4.2 and 5.4.3 outline alternative options for influencing the completion time. While the first option focuses on a short completion time, the second option supports a deadline within which a batch of tasks is supposed to be completed.

5.4.1. Motivation

The objective of the CSP/DVM is to guarantee a well-defined result quality at minimum worker effort. For each task type, there is a pool of workers who are eligible to work on assignments of the appropriate tasks. All workers who pass a certification test are added to that worker pool. Even though there is no assumption about the concrete size of the worker pool or the distribution of the worker failure rates, it can be stated that a higher minimum failure rate would result in a lower QM effort: The CSP-1 process of good workers will run less frequently in full inspection mode so fewer inspections will be required. Moreover, votes of good workers will result in fewer iterations of the DVM. Therefore, in order to really minimize the QM effort, only the very best workers should be allowed to work on the tasks.

This, however, would come with a drawback as it would compromise the scalability of the service because it would almost serialize the task execution. For a batch of tasks, the completion time would drastically grow. Yet, increasing performance by adding less qualified workers to the pool comes at the price of additional redundant submissions that are needed to compensate for the less reliable results. Most use cases, though, will likely require both: A minimum level of result quality and a maximum completion time, i.e. a deadline by which all tasks must be completed.

5.4.2. Maximum throughput

The *maximum throughput* option basically uses all available workers who have successfully completed the qualification test. As the failure rates of the workers may change over time, even those who met the test may fall below the limit at a later point in time. Therefore, the failure rates should be tracked over time and low performing workers should be removed from the worker pool. In order to do so, a maximum failure rate p_{max} can be introduced. If a worker's failure p_j rate exceeds the maximum failure rate, i.e. $p_j > p_{max}$, he will not be allowed to work on assignments of the specific task type any more. As described in the following section, this decision does not necessarily have to be irreversible, but the worker might be added back to the worker pool in times of high workload.

5.4.3. Fixed deadline

This section proposes a completion time management mechanism for the CSP/DVM that dynamically adjusts the size of the worker pool in order to finish a batch of tasks within a given deadline. The mechanism is only outlined here and will not be assessed in the evaluation in chapter 8.

The mechanism starts with a low value of the maximum failure rate p_{max} in order to start with the top performers only. By monitoring the execution progress, the expected completion time is being constantly projected. If the process is expected to finish too late, p_{max} is increased step by step in order to add more and more (less qualified) workers to the pool.

The completion time management process is intended to ensure that in the order of ascending failure rates, only the best workers are added to the worker pool which keeps the QM overhead minimal. At the same time, enough workers are being added in order to meet the deadline.

The basic idea is to check the progress in regular time intervals and forecast future productivity based on progress during the past intervals. If the projected completion time exceeds the deadline, the size of the active worker pool is increased. Example: if after half of the time only one quarter of the tasks have been completed, the size of the worker pool is tripled.

5.5. Model application

This section describes the individual activities that are required in order to apply the CSP/DVM model. As a first step, a number of parameters need to be configured for the CSP-1 and the DVM (section 5.5.1). Second, a mechanism needs to be setup to identify the initial failure rates of the contributing workers (section 5.5.2). Third, the CSP/DVM process flow described in figure 5.3 needs to be implemented (section 5.5.3). In a fourth step, a completion time management option should be selected (section 5.5.4).

5.5.1. Configuration of the CSP-1 and the DVM

As indicated in section 5.2.3, the CSP-1 is used with the extension of imperfect inspection. Because the workers are acting independently of each other, the sampling process is performed at worker level. The same $AOQL$ is applied to all workers who work on the same type of task, i.e. the same response quality is requested from all participating workers. The CSP-1 and DVM parameters need to be calculated according to the following steps:

1. Choose a value for $AOQL$: set it to the highest average amount of incorrect results that is acceptable for the requester, e.g. a value of $AOQL = 0.05$ means that there is supposed to be a maximum average of 5 percent incorrect results in the result stream.
2. Define an inspection quality level $e_1 = e_2 = \varphi_{min} > AOQL$. Note that it will influence both the calculation of the CSP-1 parameters as well as the behavior of the DVM. As a conservative configuration, the recommendation is to set both the type 1 and the type 2 inspection error to the minimum inspection quality φ_{min} delivered by the DVM. The inspection quality level should be rather high in order to allow for a fair assessment of the workers.
3. Specify the expected incoming level of response quality $\hat{p}$ of the workers. This value refers to the average failure rate of the contributing workers. The initial worker failure rates discussed in section 5.5.2 can be used to calculate this average.
4. Determine the clearance number i according to section 4.4.1 or select i depending on the typical number of tasks that a worker is expected to complete in a row. For example, it does not make sense to have a clearance number of $i = 20$ if a worker only completes 20 tasks in a row because the worker would be always in the 100% inspection phase.

5. Calculate the corresponding sampling fraction f using (4.6) by inserting the values of $AOQL$, e_1, e_2, $\hat{p}$ and i identified above.
6. Define an initial value for the escalation limit ε_{min}. It should be set to the approximate portion of tasks that are supposed to be escalated. If the complexity of the tasks does not change over time, a value of $\varepsilon_{min} = 0.01$ would result in about 1% of the tasks being escalated. The actual percentage may differ as the complexity of tasks varies. Over time, the escalated tasks and the corresponding worker responses should be analyzed in order to decide whether a corrective action should be taken, e.g. to improve the task instructions or to increase the qualification requirements. If it turns out that a large portion of tasks gets escalated by mistake, the escalation limit should be decreased. If no or just few tasks are escalated, the escalation limit should be increased, because the analysis of escalated tasks provides a good opportunity for a continuous improvement of the overall process.

5.5.2. Initialization of worker failure rates

For applying the QM model, initial values for the failure rates p_j of all workers are required. They can be determined by performing a qualification test based on a series of tasks for which the expected results are already known. There is no need to have workers assigned to the worker pool before starting the actual process because the qualification test can also be requested on the fly when the worker first decides to work on a specific task type. Only such workers that initially meet a correctness level close to AOQL should be allowed to work on the task by adding them to the worker pool. Workers that do not meet the quality needs would continuously stay in full inspection mode and, therefore, would lead to high costs.

Note that according to 5.3.1, the worker error rates must not be exactly 0 or 1 even if a worker completes all tasks of the qualification test or does not complete any of them successfully. An initial failure rate of 0 or 1 would compromise the calculation of φ_k. Because of the nature of human work it is obvious that failure rates of 0 and 1 are no realistic estimates for the future performance of a worker. Therefore, if a worker successfully completes all tasks of the test at least one of the test results should be counted as a defect. For example, in a test with 20 tasks, the minimum possible failure rate to achieve would be $p = \frac{1}{20} = 0.05$. If a worker does not complete any of the tasks successfully, he should anyway not be allowed to work on tasks (refer to section 5.5.4).

5.5.3. Detailed process flow

After calculating the CSP-1 and DVM parameters and setting up a mechanism for determining the initial worker failure rates, the QM process illustrated by figure 5.3 can be applied. The following steps describe the process flow of the QM component in detail:

1. Receive task from requester and pass a first assignment of it to the cloud labor platform.
2. Receive response along with the anonymous ID of the worker who has performed the task.
3. Based on the ID, check status of the worker's CSP-1. If the CSP-1 is in full inspection mode, the worker's response needs to be inspected, otherwise decide based on sampling fraction f whether to inspect the response. If no inspection is needed, continue with step 9.

4. Pass an additional assignment of the task to the platform.
5. Receive response along with worker ID.
6. Based on all available responses and the historical failure rates of the involved workers, calculate the probability φ_k with (5.10) or (5.11) for all available responses and identify the response d with the highest probability of correctness φ_d along with the appropriate escalation limit ε_d (5.14).
7. If the probability φ_d equals or exceeds the predefined minimum inspection quality φ_{min}, perform the following activities:
    - Update the failure rates and the CSP status of all involved workers depending on whether their response matches d or not.
    - If a response matches d and the worker is in full inspection mode, increase the corresponding worker's *defect-free-counter* i_j by one. If it reaches the clearance number i, i.e. $i_j \geq i$, switch the CSP status of the worker to fractional inspection mode.
    - If a response does *not* match d, switch the CSP status of the corresponding worker to full inspection mode.
    - Finally, continue with step 9.
8. If the current escalation limit ε_d is falling below the specified escalation limit ε_{min}, escalate the task back to the requester and proceed with the next task in step 1.
9. Accept response d as the final result and return it to the requester. Then proceed with the next task in step 1.

5.5.4. Selection of completion time management option

Finally, one of the completion time management options described in section 5.4 needs to be selected. As a starting point, the *maximum throughput option* is recommended in order to limit the implementation overhead. The maximum failure rate p_{max} may be set to the same limit as when setting up the initial worker pool in section 5.5.2. If there is a large number of workers available and there is plenty of time for completing the tasks, the fixed deadline option may be considered.

5.6. Small worker pools

When there are only few workers available, the CSP/DVM can run into situations in which voting steps are delayed due to a lack of workers. After elaborating on this challenge in section 5.6.1, section 5.6.2 introduces approaches for addressing it.

5.6.1. Motivation

An important reason for employing QM to cloud labor services is that the worker performance may change over time. Even if a worker has delivered good results for a long period of time, his performance may suddenly decline. The CSP/DVM addresses this objective by continuously tracking and updating each worker's failure rate. An update is initiated when the CSP-1 requests an inspection of a worker's response, which will issue a dynamic voting process. Once the DVM finishes, the failure rates of the involved workers are updated. However, as long as the DVM has not finished, the entire QM mechanism continues to use the original failure rates even if the actual worker performance may have declined. This delayed completion of the DVM compromises the CSP: If the result of the inspection does not become promptly available, the CSP may not react quickly enough to increasing worker error rates by switching to full inspection mode. Therefore, incorrect responses may slip through and may compromise the average response quality.

The problem does not appear if there are enough workers available to work on the tasks. Given that the iterations of the DVM are performed sequentially and assuming that all workers need a similar amount of time for completing a task, the maximum number of tasks that a worker may be ahead approximately matches the number of iterations until the DVM comes to a decision. This can typically be assumed to be a single digit number.

If, however, the pool of available workers is small, the delay can be much longer. In the worst case, a single available worker may perform assignments of all available tasks without any of the inspection requests being completed and without the CSP status of the worker being updated ever. In order to avoid that, additional measures have to be taken.

5.6.2. Prevention of delayed task inspection

As discussed, the timing of the inspection process is essential in the CSP/DVM scenario. In this section, three options are presented that reduce or avoid the effect of delayed task inspection.

Finish incomplete tasks first

Incomplete tasks are defined to be such tasks for which the DVM is in process but did not finish yet. A general best practice is to prioritize the assignments of such tasks higher than the ones of new tasks: If the CSP-1 has identified that inspection is required for a task, the assignments generated in the course of the inspection should be put at the top of the queue so that workers will grab them first before working on so far untouched tasks. As long as enough workers are available, this procedure ensures that incomplete tasks are finished before new ones. MTurk implicitly supports this behavior.

Of course, each worker anyway only sees assignments of such tasks that he is eligible to work on. In particular the worker will not see assignments of tasks he has already worked on even if they are positioned at the top of the queue. Therefore, in order to complete all inspections in a timely manner, the number of different workers must match at least the maximum number of iterations in the DVM. If there are fewer workers, at least some inspection processes will get stuck.

Limit number of incomplete tasks

In order to avoid situations in which too many inspections are getting stuck, the number of incomplete tasks may be limited either per worker or in general. By limiting it per worker, such workers who have already contributed to a certain number of incomplete tasks are prevented from working on new tasks until at least one of the incomplete ones has been completed. Other workers may still continue to do work. When applying a general limit, all workers are prevented from doing work once a certain number of undecided inspections has been reached.

Naturally, if there is a lack of workers, both approaches sooner or later lead to an interruption of the task execution process. The general limit will typically be reached earlier but will therefore be easier to implement and to monitor.

Full inspection

A simple but very effective approach to avoid the issues is to run the CSP-1 in *full inspection* mode which is equivalent to not using it at all. If the CSP-1 is not used, a timely inspection is not needed.

The full inspection mode has been discussed in section 5.6.1. It can be turned on by simply setting the clearance number i to a value that exceeds the number of expected tasks or by setting the sample fraction f to 1. As a consequence, the DVM is issued for each individual task. The desired output quality is then defined by the minimum inspection quality level φ_{min} of the DVM rather than by $AOQL$.

The full inspection mode can be assumed to increase the inspection costs because each task is performed at least by two workers while a portion of tasks is only performed by a single worker when using the CSP-1. In return, because of the lower value of the minimum inspection quality φ_{min}, the average number of iterations of the DVM is reduced.

Like the incomplete task limit, full inspection will not avoid the process from getting stuck if there are too few workers. However, with full inspection only those tasks will be left over that need more workers for completion.

6. Model variations

The model variations introduced by this chapter complement the core model presented in the previous chapter by addressing a broader set of relevant cloud labor scenarios. The considerations provide the basis for two of the case studies presented in chapter 9.

While in section 6.1, the dynamic voting mechanism (DVM) is adapted to the important class of *multi-labeling* scenarios, section 6.2 introduces a complementary model that makes SQC applicable to *non-deterministic* tasks which are not supported by the DVM.

6.1. Multi–labeling scenarios

Besides the generation of content, the collection of data and the transcription of recorded speech, one of the most popular types of cloud labor tasks is the classification or categorization of items (Ipeirotis, 2010a). In many of such scenarios, multiple labels must be assigned to each item at the same time. Such *multi-labeling* scenarios are highly relevant in research and practice. Basic examples comprise the tagging of texts and multimedia content according to specific characteristics and the classification of medical symptoms and diseases. The most prominent application of multi-labeling is probably the *ESP Game* that was introduced in section 2.2.

In its generic form, the DVM is not suited to manage the quality of multi-labeling tasks. The aim of this section is to develop a variation of the approach which closes this gap.

After providing an overview on the general types of classification scenarios in section 6.1.1 and introducing the scenario assumptions in section 6.1.2, section 6.1.3 explains how the DVM has been applied to the multi-labeling scenario. The basic process flow of the model is then described in section 6.1.4. Sections 6.1.5 to 6.1.7 cover the actual statistical model.

6.1.1. Types of classification scenarios

This section provides an overview on the general types of classification scenarios following the terminology used by Tsoumakas & Katakis (2007).

In basic classification scenarios, items are classified by a single label which is taken from a set of disjoint labels. If the set comprises just two different labels, one speaks of a *binary* *classification.* The inspection mechanism of sampling plans for attributes represents such a classification as it just differentiates between good and bad items. If an item is supposed to be classified according to a set of more than two disjoint labels, this would be a *multi-class classification.* An example would be to categorize pictures according to the countries where they were taken. More complex classification scenarios require more than one label to be assigned to the item at the same time, still taken from the same disjoint set of labels. For instance, when classifying pictures according to the objects they show, there may obviously be more than one label assigned to an image. Such tasks are called *multi-label classifications.*

Table 6.1 summarizes the classification types introduced above. The second column indicates the number of labels that are available, the third column indicates the number of labels that are assigned to an item. The specific characteristic of multi-labeling classifications is that multiple labels are available and any number of them can be assigned to each item. If the available labels represent a hierarchy such that one or more nodes of the hierarchy are assigned to an item, the corresponding task is called *hierarchical* multi-label classification, otherwise it is a *flat* multi-label classification. The model introduced in the following focuses on the latter type.

Table 6.1.: Important types of classification scenarios.

Type of classification	Number of labels	Labels assigned	Example scenario
Binary classification	$l = 2$	1	Classify samples into good and bad items.
Multi-class classification	$l > 2$	1	Categorize pictures according to the countries where they were taken.
Multi-label classifications	$l > 1$	$0, 1, 2, .., l$	Classify pictures according to the objects they show.

6.1.2. Assumptions and definitions

In addition to the assumptions of the generic DVM described in section 5.2.1, specific assumptions about the multi-labeling scenario are made. It consists of a series of labeling tasks, in each of which a predefined set of binary labels is assigned to an item.

The set $B = \{b_1, b_2, .., b_v\}$ represents the available labels. While in the generic DVM in section 5.3.1 each worker $y = 1, .., w$ is assumed to return a single $x \in A$, in the multi-labeling scenario discussed here, a worker can assign multiple labels to an item. A decision needs to be made for every label in the set B. All labels that a worker does not explicitly assign to an item are considered to be denied. This corresponds to the "closed topic assumption" defined by Sasaki et al. (2007). The labels are assumed to be statistically independent of each other. In particular, they do not have a hierarchical relationship and are not composed of multiple sub-labels but represent a flat multi-label classification.

The labels are also not ranked or scored and the decisions of the workers are independent. Furthermore, a fixed cost per worker and labeling-task is assumed. Another important assumption is that the challenge of assigning a correct label is similar for all labels, i.e. there are assumed to be no labels which are extraordinarily difficult to adjudicate upon.

The responses from the workers $y = 1, .., w$ regarding the labeling decisions for labels $x = 1, .., v$ are represented by a response matrix $\tilde{R} = (r_{xy})$:

$$\tilde{R}_{vw} = (r_{xy}) = \begin{bmatrix} r_{11} & \cdots & r_{1w} \\ \vdots & r_{xy} & \vdots \\ r_{v1} & \cdots & r_{vw} \end{bmatrix} \tag{6.1}$$

The elements $r_{xy} \in \{0, 1\}$ of the matrix indicate if label x has been assigned ($r_{xy} = 1$) or has not been assigned ($r_{xy} = 0$) by worker y. The ground truth is unknown. The (unknown) true binary value, for each label x is represented by an indicator vector $\overrightarrow{t} = (t_1, .., t_v)$ with $t_x \in \{0, 1\}$. The value $t_x = 1$ indicates that label x should be assigned; $t_x = 0$ in turn indicates that label x should not be assigned. Table 6.2 illustrates the possible combinations between the worker's labeling decision r_{xy} and the true value of the label $t_x = 1$. The *true positives* (TP) define the number of *positive* labels rightly assigned by a worker while the *true negatives* (TN) define the number of *negative* labels rightly omitted by the worker. The *negative* labels wrongly assigned by a worker are called *false positives* (FP) while the number of *positive* labels wrongly omitted by a worker are called *false negatives* (FN).

Table 6.2.: Confusion matrix indicating the possible combinations of worker decisions and true labels.

		True label t_x	
		1	0
Worker decision r_{xy}	1	True positive (TP)	False positive (FP)
	0	False negative (FN)	True negative (TN)

The sensitivity $\alpha_y \in [0, 1]$ is defined as the proportion of correctly identified positive labels (also called *true positive rate*) and can be understood as the conditional probability that a worker assigns a label ($r_{xy} = 1$) given it should be assigned according to its true value ($t_x = 1$). In can be calculated based on the number of TPs and FNs:

$$\alpha = \frac{TP}{TP + FN} \tag{6.2}$$

The specificity $\beta_y \in [0, 1]$ is defined as the proportion of correctly identified negative labels (also known as *true negative rate*) and can be understood as the conditional probability that a worker does not assign a label ($r_{xy} = 0$) given that the label should not be assigned according to its true value ($t_x = 0$). It can be calculated based on the number of TNs and FPs:

$$\beta = \frac{TN}{TN + FP} \tag{6.3}$$

6.1.3. Application of the DVM

According to the taxonomy introduced in section 3.3.1, mutli-labeling is a *fine-grained* task because it could be split into smaller ones, e.g. one task per label.[1] The assumption of a ground truth $\overrightarrow{t}$ representing the true labels implies that the multi-labeling task is also

[1] If there is a large number of tasks $b \in B$, this would obviously not be efficient.

deterministic. For managing the quality of fine-grained deterministic tasks, the decision matrix in section 3.4.1 recommends to apply the iteration pattern but also mentions that the voting pattern may be a viable alternative for specific tasks. This section illustrates why the general DVM is only of limited use for multi-labeling scenarios and how it can be adjusted in order to close the gap.

A basic way to apply the DVM would be to treat any combination of possible labels $b \in B$ as a separate response $a \in A$ in section 5.3.1. By taking into account the historic failure rates of the workers, the DVM would then identify the response with the highest probability of correctness and would return it to the requester if it meets the quality needs. However, this only works if multiple users can be expected to return exactly the right combination of labels, in other words, if the entire labeling task is *simple* according to the definition in section 3.3.1.

If there is a large number of labels, getting all of them right will be difficult even for experienced workers. Therefore, just selecting one of the combinations returned by a specific user will not work. Instead, the objective of the DVM for multi-labeling is to *merge* labeling decisions from multiple workers in a way that they best match the requester's quality objectives. The final labeling decision may be a mixture of labels identified by different workers.

This variation of the DVM does not apply acceptance sampling as for merging responses from multiple workers, at least two responses are needed. Furthermore, no escalation mechanism is provided which is rather a limitation of the current model than a general restriction introduced by the multi-labeling scenario. Another significant difference compared to the general DVM concerns the way the requester's quality objectives as well as the historical worker performance are represented. A single failure rate is not an adequate measure for the multi-labeling scenario. Instead, the sensitivity and specificity measures defined in section 6.1.2 are used. The quality objectives are defined by the minimum sensitivity α_{min} and specitivity β_{min}. The historical performance of worker y is defined by his sensitivity α_y and specificity β_y.

In the following section, the process flow of the DVM for multi-labeling will be described.

6.1.4. Process flow

The basic process flow of the DVM for multi-labeling is illustrated by figure 6.1. While the platform layer and the worker pool at the bottom of the diagram are identical to the general DVM process flow in figure 5.3, the QM component and the requester layers differ in the following points which have been motivated in the previous section:

1. No acceptance sampling is used (no CSP-1).
2. There is no escalation mechanism.
3. Worker performance is measured by sensitivity and specificity rather than by a single failure rate.

The process flow is as follows: After receiving a labeling task from the service requester along with the quality objectives, two assignments of the task are immediately published to the cloud labor platform.[2] After receiving the worker's labeling decisions along with

[2] In favor of a simplified diagram, the process flow suggests that the second assignment of the task is only published after having received the response of the first one. However, both assignments are actually published immediately.

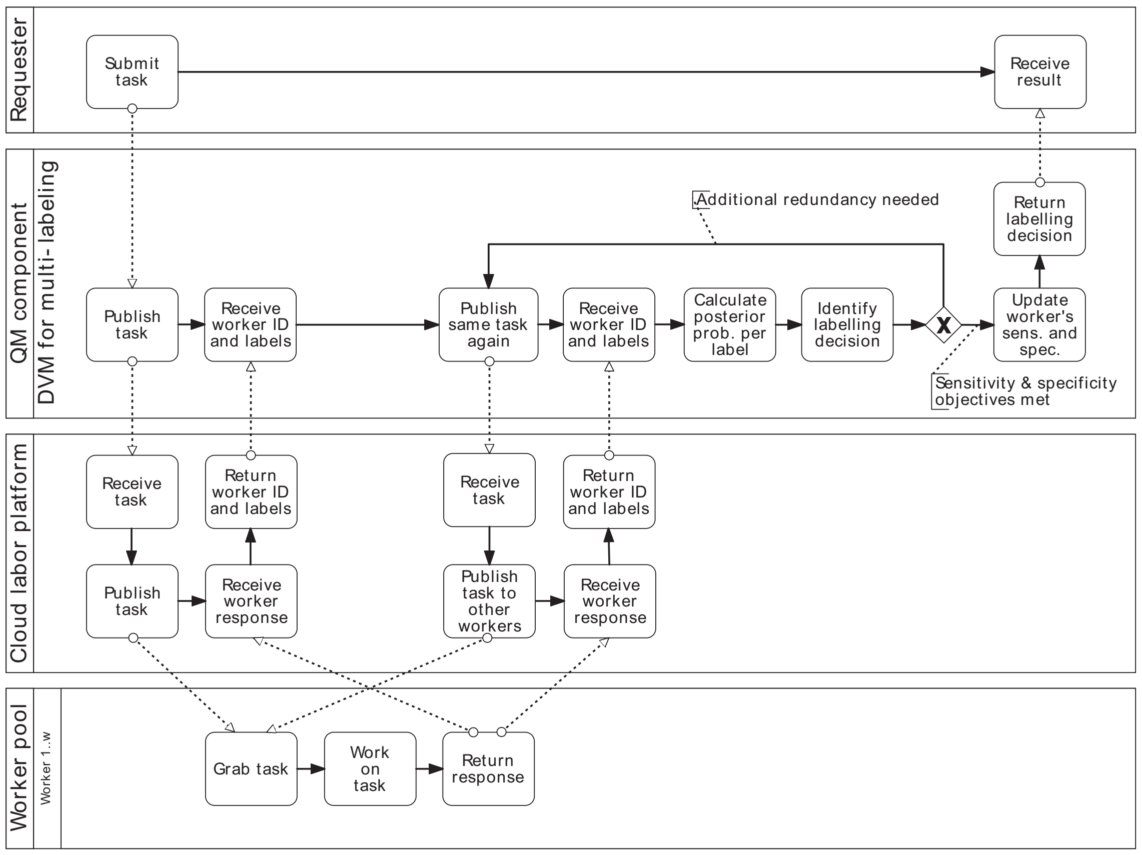

Requester
Submit task
Receive result
QM component
DVM for multi- labeling
Additional redundancy needed
Return labelling decision
Publish task
Receive worker ID and labels
Publish same task again
Receive worker ID and labels
Calculate posterior prob. per label
Identify labelling decision
Update worker's sens. and spec.
Sensitivity & specificity objectives met
Cloud labor platform
Receive task
Return worker ID and labels
Receive task
Return worker ID and labels
Publish task
Receive worker response
Publish task to other workers
Receive worker response
Worker pool
Worker 1..w
Grab task
Work on task
Return response

Figure 6.1.: Process flow of the DVM for multi-labeling.

their worker IDs, the posterior probability of the individual labels is calculated on behalf of the sensitivity and specificity values of the contributing workers. In a next step, the best suitable labeling decision is identified. If the latter one already meets the requester's quality objectives, the sensitivity and specificity of the contributing workers are updated and the consolidated labeling decision is returned to the requester. If the quality objectives are not yet met, another instance of the same task is published to the cloud labor platform and the loop continues.

The crucial steps are explained in the following sections: Section 6.1.5 describes how the posterior probability per label is calculated, 6.1.6 covers the identification of the best suitable labeling decision and 6.1.7 addresses the update of the worker's sensitivity and specificity values.

6.1.5. Posterior probability per label

This section explains how the posterior probabilities for assigning each individual label are estimated based on the labeling decisions of the workers. Following Warfield et al. (2004), he posterior probability $P(t_x = 1|r_1, .., r_w)$ of a label x having the true value $t_x = 1$ given the observed labeling decisions of w workers can be calculated using Bayes' theorem: The posterior probability after observing the labeling decisions $r_x \in \{0, 1\}$ can be computed as:

$$LP_x \equiv P(t_x = 1|r_{x1}, .., r_{xw}) = \frac{P(t_x = 1, r_{x1}, .., r_{xw})}{P(r_{x1}, .., r_{xw})} \tag{6.4}$$

$$= \frac{P(t_x = 1) \cdot P(r_{x1}, .., r_{xw}|t_x = 1)}{(P(t_x = 1) \cdot P(r_{x1}, .., r_{xw}|t_x = 1) + (1 - P(t_x = 1)) \cdot P(r_{x1}, .., r_{xw}|t_x = 0))} \tag{6.5}$$

Because the labels are assumed to be binary, the probability for a label being assigned must be one minus the probability for the label being unassigned, i.e.

$$P(t_x = 0) \equiv 1 - P(t_x = 1). \tag{6.6}$$

Furthermore, because of the assumption of independent labels and workers, the following transformations can be performed:

$$P(r_{x1}, .., r_{xw}|t_x = 1) \equiv \prod_y P(r_{xy}|t_x = 1) \tag{6.7}$$

$$P(r_{x1}, .., r_{xw}|t_x = 0) \equiv \prod_y P(r_{xy}|t_x = 0) \tag{6.8}$$

With (6.6) through (6.8), (6.4) turns into

$$LP_x = \frac{P(t_x = 1) \cdot \prod\limits_y P(r_{xy}|t_x = 1)}{P(t_x = 1) \cdot \prod\limits_y P(r_{xy}|t_x = 1) + (1 - P(t_x = 1)) \cdot \prod\limits_y P(r_{xy}|t_x = 0)}. \tag{6.9}$$

With the definitions of α and β from (6.2) and (6.3) and with the definition of the prior probability for a label being true.

$$\tilde{p} \equiv P(t_x = 1) \tag{6.10}$$

this finally leads to:

$$LP_x = \frac{P(t_x = 1) \cdot \prod\limits_{y|r_{xy}=1} \alpha_y \prod\limits_{y|r_{xy}=0} (1 - \alpha_y)}{\tilde{p} \cdot \prod\limits_{y|r_{xy}=1} \alpha_y \prod\limits_{y|r_{xy}=0} (1 - \alpha_y) + (1 - \tilde{p}) \cdot \prod\limits_{y|r_{xy}=0} \beta_y \prod\limits_{y|r_{xy}=1} (1 - \beta_y)} \tag{6.11}$$

The prior probability $\tilde{p}$ typically has to be estimated. The initial sensitivity and specificity of the workers can be calculated with (6.2) and (6.3) based on the results of a qualification test. However, as mentioned by Warfield et al. (2004) and others, the initial values for sensitivity and specificity should never be 1 as this would not be a realistic estimate for the future performance of a worker. Therefore, it is recommended to assume that at least one FP and one FN was returned even if a worker actually passes the qualification test without errors.[3]

6.1.6. Identification of best suitable labeling decision

In order to check whether the requested correctness goal in terms of the minimum sensitivity α_{min} and specitivity β_{min} is met, the following aggregation mechanism is used: For each relevant combination of true and false labels, the projected sensitivity and specitivity

[3] Compare to the section 5.5.2.

is calculated using the posterior probabilities determined in (6.11). The combination that best meets the objectives is returned to the requester. If there is no such combination, the redundancy is increased by requesting feedback from another worker.

It is actually not necessary to calculate the expected sensitivity and specitivity for all permutations of true and false labels because it is obvious that most permutations are irrelevant. For example, if there is only one label in the final result, it will certainly be the one with the highest probability of correctness $LP_j|\forall l \leq v : LP_j \geq LP_l$. If there is a second one, it will be the one with the second highest probability and so on. This will become clear based on the following considerations.

Let the tuple H contain the indices of all labels sorted by their probability of correctness LP:

$$H = (h_1, h_2, .., h_v) \text{ with } LP_{h_1} \geq LP_{h_2} \geq \cdots \geq LP_{h_v} \tag{6.12}$$

The relevant combinations of labels only differ in the number $k \leq v$ of labels assigned. Let further $\overline{TP}_k$, $\overline{FN}_k$, $\overline{FP}_k$ and $\overline{TN}_k$ represent the projected average number of TPs, FNs, FPs and TNs when returning the k labels with the highest probability of correctness. $\overline{TP}_k$ and $\overline{FN}_k$ can be calculated as

$$\overline{TP}_k = \begin{cases} 0 & \text{if } k = 0 \\ \sum\limits_{y=1}^{k} LP_{h_y} & \text{if } k > 0 \end{cases} \tag{6.13}$$

$$\overline{FN}_k = \begin{cases} 0 & \text{if } k = 0 \\ \sum\limits_{y=1}^{k} (1 - LP_{h_y}) & \text{if } k > 0 \end{cases} . \tag{6.14}$$

The remaining $v - k$ labels are not returned even though there is a non-zero probability that they are correct. Therefore, $\overline{FN}_k$ and $\overline{TN}_k$ can be calculated as

$$\overline{FP}_k = \begin{cases} 0 & \text{if } k = v \\ \sum\limits_{y=k+1}^{v} LP_{h_y} & \text{if } k < v \end{cases} \tag{6.15}$$

$$\overline{TN}_k = \begin{cases} 0 & \text{if } k = v \\ \sum\limits_{y=k+1}^{v} (1 - LP_{h_y}) & \text{if } k < v \end{cases} . \tag{6.16}$$

Using (6.13) through (6.15), the projected sensitivity $\overline{\alpha}_k$ and specitivity $\overline{\beta}_k$ can be calculated according to (6.2) and (6.3) with $k > 0$ being the projected number of labels to be assigned and $v - k > 0$ being the projected number of labels not to be assigned:

$$\overline{\alpha}_k = \frac{\overline{TP}_k}{\overline{TP}_k + \overline{FN}_k} = \frac{\sum\limits_{x=1}^{k} LP_{w_x}}{\sum\limits_{x=1}^{k} LP_{w_x} + \sum\limits_{y=k+1}^{v} (1 - LP_{h_y})} \tag{6.17}$$

$$\overline{\beta}_k = \frac{\overline{TN}_k}{\overline{TN}_k + \overline{FP}_k} = \frac{\sum\limits_{x=k+1}^{v} (1 - LP_{w_x})}{\sum\limits_{x=k+1}^{v} (1 - LP_{w_x}) + \sum\limits_{y=k+1}^{v} LP_{h_y}} \tag{6.18}$$

Figure 6.2 illustrates the calculation of $\overline{TP}$, $\overline{FN}$, $\overline{FP}$ and $\overline{TN}$ for the case $v = 5$ and $k = 3$, i.e. the three labels with the highest values of LP are assumed correct. Each bar is representing a label. The lower (dark grey) portion of the bar indicates its labeling probability LP, while the upper (light grey) portion indicates $(1 - LP)$. Assuming that the first three labels are correct, the projected average number of TPs is equivalent to the combined dark grey areas of the bars 1 to 3. The combined light grey areas of these bars are equivalent to the projected average number of FNs. Accordingly, the combined dark grey areas of the remaining two bars 4 and 5 represent the projected average number of FPs while the combined light grey areas of these bars represent the projected average number of TNs.

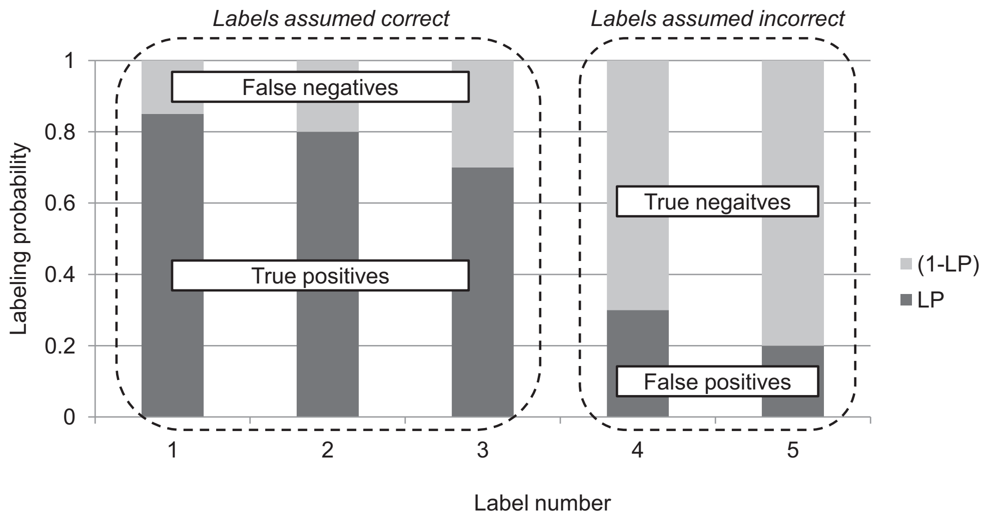


Figure 6.2.: Illustration of the calculation of $\overline{TP}$, $\overline{FN}$, $\overline{FP}$ and $\overline{TN}$ based on the labeling probability LP.

The parameter k is increased step by step until both the sensitivity and specificity goals are met, i.e. until the projected sensitivity and specificity are at least equal to the requested ones, i.e. $\overline{\alpha}_k \geq \alpha_{min}$ and $\overline{\beta}_k \geq \beta_{min}$. As $\overline{\alpha}_k$ is not defined for $k = 0$ it is assumed that the sensitivity objective is met. Accordingly, for $k = v$, it is assumed that the specificity objective is met. As a consequence, if no label is returned by any of the workers, the DVM for multi-labeling accepts this as the consolidated worker decision and does not return any label to the requester.

6.1.7. Updating the worker's sensitivity and specificity

After successfully completing a task, the performance indicators of all w participating workers are being updated. In (6.19) and (6.20), the sensitivity α_y is calculated as the sum of the posterior probability of all labels the worker submitted divided by the sum of posterior probabilities of all labels with $x = 1, .., v$ denoting all possible labels and $P(t_x = 1|r_{x1}, .., r_{xw})$ denoting the posterior probability of label x being correct given the observed labeling decisions of w workers:

$$\alpha_y = \frac{\sum\limits_{x|r_{xy}=1}^{v} P(t_x = 1|r_{x1}, .., r_{xw})}{\sum\limits_{x=1}^{v} P(t_x = 1|r_{x1}, .., r_{xw})} \tag{6.19}$$

The individual worker's specificity β_y is calculated accordingly:

$$\beta_y = \frac{\sum\limits_{x|r_{xy}=0}^{v} (1 - P(t_x = 1|r_{x1}, .., r_{xw}))}{\sum\limits_{x=1}^{v} (1 - P(t_x = 1|r_{x1}, .., r_{xw}))} \tag{6.20}$$

The DVM for multi-labeling will be further discussed in section 9.1 where it will be applied to a medical coding scenario.

6.2. Non-deterministic tasks

As anticipated in section 5.1, the *group validation approach* introduced in this section[4] is not an enhancement of the DVM but represents a complementary approach that relies on a different QM pattern, a different SQC mechanism and a different method for sample inspection. The objective of the group validation approach is to approach the research question of this thesis in an alternative way in order to gain a broader view on SQC for cloud labor services. At the same time, it aims to overcome the restriction of the DVM to be limited to deterministic tasks.

After motivating the group validation approach in section 6.2.1 and providing its statistical foundation in section 6.2.2, section 6.2.3 introduces the process flow of the approach. Section 6.2.4 and 6.2.5 then describe the sampling mechanism and the sample inspection mechanism in detail. Finally, section 6.2.6 discusses how to determine the number of reviewers to be used for the sample inspection.

6.2.1. Introduction

According to the discussion in section 5.1, the *validation* pattern represents another promising approach to be used in combination with SQC. The comparison of QM patterns in section 3.4.1 has revealed that it is an extremely flexible approach that can be applied with little effort to a wide range of scenarios. Moreover, it intuitively fits well into SQC because the concept of validation is close to that of inspection which can be seen as the core constituent of SQC. However, feedback from multiple reviewers may need to be combined in order to come to a reliable review decision. Therefore, the group validation approach presented here relies on a vote of multiple reviewers.

As opposed to the DVM approach, the number of reviewers is not iteratively increased, but a fixed number of validation decisions is allocated per task. For a batch of tasks, the validation decisions are then consolidated using a maximum likelihood estimation, which has been successfully applied to a series of cloud labor scenarios, e.g. Raykar et al. (2009) and Ipeirotis et al. (2010), but not yet in the context of the validation pattern. An important characteristic of the approach is that it is batch oriented, therefore it can be seen as the perfect counterpart to the concept of acceptance sampling, which is batch oriented as well. By combining the group validation approach with the Dodge-Romig AOQL sampling plan, a *sample-based group validation* approach is designed. While the EM algorithm and the Dodge-Romig AOQL sampling plan are state-of-the-art approaches,

[4] Considerable parts of this section have already been exposed to and tested with the academic community, having been presented at the 16th Americas Conference on Information Systems (AMCIS 2010) and published in the respective proceedings (Kern et al., 2010e).

their specific combination into a QM mechanism for cloud labor services represents a contribution of this thesis. In fact, the group validation approach is the first application of acceptance sampling for cloud labor services.

The underlying assumptions are consistent with the ones presented in section 5.2.1, except that the approach is no longer limited to deterministic tasks but also applies to non-deterministic ones.

6.2.2. Fundamentals

This chapter describes the EM algorithm that was outlined in section 3.3.3. It was originally developed by Dempster et al. (1977) who introduced it as a generalized method to compute maximum likelihood estimates from incomplete data. Dawid & Skene (1979) first applied it to a scenario in which a subject is classified into one of multiple categories based on the judgments of multiple observers. In addition to an estimate for the true category of the subject, the method also derives error rates for the observers. The basic idea of the algorithm is to infer the posterior probability of the subject's true category by integrating its a priori probability, the observers' judgments and the observers' error rates via the Bayes Theorem. This is done in an iterative way until the posterior probabilities converge. The approach proposed by Dawid & Skene is outlined in the following:

In an experiment, w independent observers $y = 1, .., w$ are asked to assign one or multiple of x possible categories $x = 1, .., v$ to a set of $z = 1, .., s$ items. The same category can be assigned multiple times, but not every observer needs to assign a category to every item. Let $n_{zx}^{(y)}$ represent the number of times that observer y states that subject z belongs to category x. The observers may perform errors which are described by the error rates $\pi_{xl}^{(y)}$, defined by

$$\pi_{xl}^{(y)} = \frac{\text{(number of cases observer y records category l when x is correct)}}{\text{(number of cases assessed by observer y where category x is correct)}}. \tag{6.21}$$

Let T_{zx} represent the probability that subject z belongs to category x. If the true category $x = t$ of subject z was known, the probability would be $T_{zt} = 1$ for the true label $x = t$ and $T_{zx} = 0$ for $x \neq t$. However, the true categories are unknown and can therefore only be estimated. The estimates can take any value $T_{zx} \in [0; 1]$.

The maximum likelihood estimates of the observer error rates can be calculated by

$$\widehat{\pi}_{xl}^{(y)} = \frac{\sum\limits_{z=0}^{s} T_{zx}}{\sum\limits_{l=0}^{v} \sum\limits_{z=0}^{s} (T_{zx} \cdot n_{zl}^{y})} \tag{6.22}$$

Note that (6.22) is semantically equal to (6.21). The probabilities of the categories $p_x(x = 1, .., v)$ can be estimated via

$$\widehat{p}_x = \sum_{z=0}^{s} \frac{T_{zx}}{s} \tag{6.23}$$

The objective of the approach is to provide estimates (T_{zx}) for the true categories along with estimates for the observer error rates $\left(\widehat{\pi}_{xl}^{(y)}\right)$.

If the probabilities (p_x) and the error rates $\left(\pi_{xl}^{(y)}\right)$ were known, Bayes' Theorem could be used to obtain estimates for (T_{zx}) using

$$P(T_{zx} = 1 \mid data) = P(data \mid T_{zx} = 1) \cdot P(T_{zx} = 1) \qquad (6.24)$$

which in numerical terms is

$$P(T_{zx} = 1 \mid data) = \frac{\prod\limits_{y=1}^{w} \prod\limits_{l=1}^{v} (\pi_{xl}^{(y)})^{n_{zl}^{(y)}} \cdot p_x}{\sum\limits_{t=1}^{v} \prod\limits_{y=1}^{w} \prod\limits_{l=1}^{v} (\pi_{ql}^{(y)})^{n_{zl}^{(y)}} \cdot p_t} . \qquad (6.25)$$

If the probabilities (p_x) and the error rates $\left(\pi_{xl}^{(y)}\right)$ are not known but estimated, the calculation can still be used and provides a maximum likelihood estimation for the probabilities (T_{zx}).

Dawid & Skene propose the following iterative procedure:

1. Obtain initial estimates for the probabilities T_{zx}.
2. Use (6.22) and (6.23) to estimate the probabilities p_x and the error rates $\pi_{xl}^{(y)}$.
3. Use (6.25) and the estimates of the probabilities p_x and the error rates $\pi_{xl}^{(y)}$ to get new estimates of the probabilities T_{zx}.
4. Repeat steps (2) and (3) until the probabilities T_{zx} converge.

If there are no good initial estimates available for the probabilities T_{zx}, the algorithm may not deliver a satisfactory result. Dawid & Skene propose to use the numbers $n_{zx}^{(y)}$ as a starting point:

$$\widehat{T}_{zx} = \frac{\sum\limits_{y=1}^{w} n_{zx}^{(y)}}{\sum\limits_{y=1}^{w} \sum\limits_{l=1}^{v} n_{zl}^{(y)}} \qquad (6.26)$$

6.2.3. Process flow

The actual QM process can be split into three phases: task execution, response validation and rework. Figure 6.3 illustrates the basic schema[5] of the group validation approach. The following sections will describe each of the phases in detail.

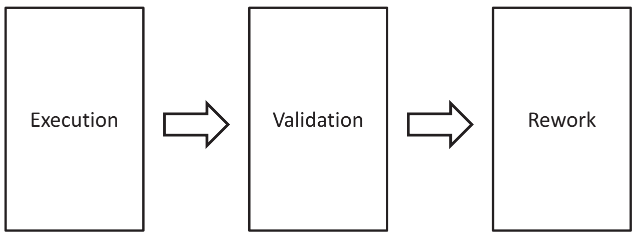


Figure 6.3.: Basic schema of the group validation approach.

[5] The complete process of the sample-based group validation is depicted in figure A.1 in appendix A.

Execution phase

The execution phase manages the generation of the responses which are then validated in the subsequent step. Figure 6.4 illustrates the steps performed in the execution phase: The batch of tasks is received from the requester (1) and passed to the cloud labor platform (2). As the responses are coming back, they are collected (3) and finally returned to the QM component along with the IDs of the workers who have performed the tasks (4).

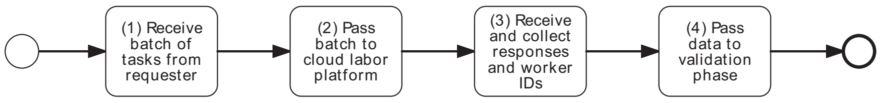

(1) Receive batch of tasks from requester
(2) Pass batch to cloud labor platform
(3) Receive and collect responses and worker IDs
(4) Pass data to validation phase

Figure 6.4.: Execution phase of the group validation approach.

Validation phase

Once all responses have been executed, the validation phase is initiated which is based on a combination of the Dodge-Romig acceptance sampling plan described in section 4.3.2 and the EM algorithm introduced in section 6.2.2. Figure 6.5 illustrates the validation phase.

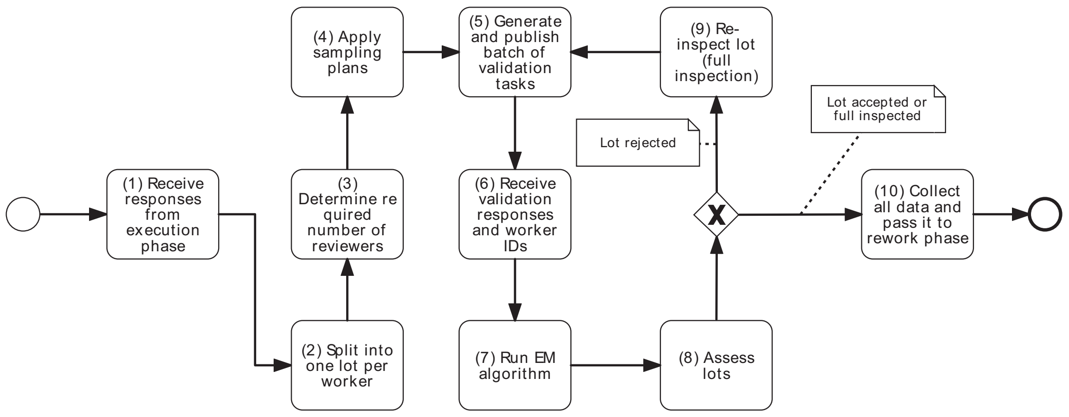

(1) Receive responses from execution phase
(2) Split into one lot per worker
(3) Determine required number of reviewers
(4) Apply sampling plans
(5) Generate and publish batch of validation tasks
(6) Receive validation responses and worker IDs
(7) Run EM algorithm
(8) Assess lots
(9) Re-inspect lot (full inspection)
Lot rejected
Lot accepted or full inspected
(10) Collect all data and pass it to rework phase

Figure 6.5.: Validation phase of the group validation approach.

First, the responses to be validated are received from the execution phase (1). Then, for the sample-based inspection, all responses received from a single worker are split into a separate lot (2). After identifying the required number of reviewers for the inspection (3), a sampling plan is applied to each lot by generating the appropriate number of validation tasks for a sample of responses from each lot (4). Afterwards the batch of validation tasks is passed to the cloud labor platform (5) where they are made available to the reviewers, who are asked to rate the responses in a binary way, i.e. to decide whether a response meets the task objectives or not. If it does not, the reviewer is asked to explain in a free text field what the issue has been with the response and how it could be corrected. After receiving the review decision along with the feedback and the reviewer's worker IDs (6), the reviewer decisions are consolidated by applying the EM algorithm to the entire batch of validation decisions for all workers (7). Then, the individual lots are assessed using the consolidated reviewer decisions (8). If a lot is rejected, a re-inspection in full-inspection mode is initiated (9) and the process continues with step (5). For such lots that have been accepted or have already been re-inspected in full-inspection mode, the reviewer decisions, the reviewer feedback and the original responses are collected and finally passed to the rework phase of the process (10).

Rework phase

The rework phase is described by figure 6.6. After the entire batch of worker responses and review decisions has arrived (1), the original worker responses are split into two partitions (2): Those that have not been inspected at all or have been inspected as ”good” are collected for return to the requester without further processing (5). Those that have been inspected ”defective” are passed back to the original workers for rework along with the reviewer’s feedback about what has been the issue with the response (3). After receiving the reworked responses (4), those are also collected (5). In a last step, the final responses are returned to the requester (6).

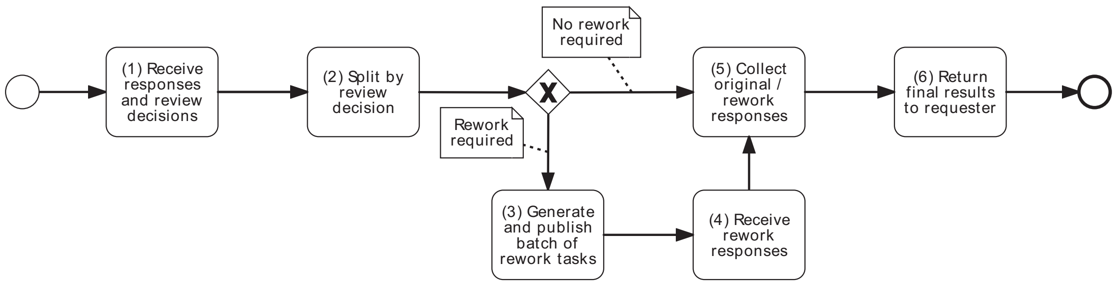

No rework required
(1) Receive responses and review decisions
(2) Split by review decision
X
Rework required
(5) Collect original / rework responses
(6) Return final results to requester
(3) Generate and publish batch of rework tasks
(4) Receive rework responses

Figure 6.6.: Rework phase of the group validation approach.

Given the fact that there is feedback available from multiple reviewers, it is assumed that the workers are able to successfully correct the issue and return an acceptable result. This assumption is even made for type I errors, i.e. if the original response had been correct and the worker is asked to rework it because of an incorrect reviewer decision, the original worker is assumed to realize the situation and keep the original response unchanged. The rationale for the assumption is that in such a situation, the reviewers would likely provide conflicting feedback which would not provide enough justification for a rework anyway. In fact, the workers should be asked to ignore the feedback as long as it is incomprehensible or there is no consistency among multiple reviewers.

The advantage of having the rework done by the original worker rather than by a different one is threefold: First, the approach will result in a learning effect for the original worker. According to section 2.6.3, feedback is a key ingredient to sustainable worker performance. Second, it will likely reduce the rework effort because the original worker is already familiar with the task. Thirdly, the approach gives the worker the chance to finally successfully complete the tasks and get paid for it. However, if the original worker is not available, the rework task is assigned to a different worker who is also assumed to provide the correct response.

After the actual rework has been finished, the two partitions of responses are merged and returned to the requester.

6.2.4. Sampling process

There are two arguments for applying acceptance sampling to the group validation process. On the one hand, acceptance sampling can reduce the inspection effort and therefore increase the efficiency, on the other hand it supports goal-based quality control according to specific quality targets. Like in the CSP/DVM model, the aim of the group validation approach is to guarantee a certain $AOQL$ while minimizing the inspection effort. In order to achieve that, the Dodge-Romig AOQL sampling plan is used. As a rectifying sampling plan it requires an error-free rework of defectives which is mentioned in the assumptions discussed in section 6.2.3. Furthermore, an error-free inspection of samples is assumed which will be discussed in section 6.2.6.

In order to meet the general requirement of a homogeneous batch of units, lots are constructed per worker. If the results need to be returned to the requester within a certain deadline, a sequence of lots may be constructed, each covering a limited time interval. In combination with the fact that a single worker may only return a small number of responses, the assumption of large lots will not always be met, which may reduce the efficiency of the approach because the inspection effort does not drop proportional to the lot size. A benefit of lot sequences is that the estimators for the EM algorithm can be continuously updated.

The actual sampling plan is being constructed according to table 4.1. Lots are accepted or rejected depending on the estimations of the EM algorithm. If the estimated number of incorrect responses exceeds the acceptance number c, the lot is rejected, otherwise it is accepted. Rejected lots will be re-inspected in 100 percent inspection mode. ATI will be minimal, if the incoming fraction defective has been estimated appropriately.

6.2.5. Inspection process

This section describes the response inspection step of the group validation approach illustrated in figure 6.5. It is based on the EM algorithm described in section 6.2.2. The cloud labor scenario meets the needs of the EM algorithm as it assumes independent workers. Furthermore, a review decision represents a judgment according to categories which is supported by Dawid & Skene (1979)'s approach.

Definitions

The following representation of the scenario assumes that there is a sample $S = \{1, .., s\}$ of worker responses $z \in S$, each being reviewed by a subset of r reviewers $y \in W$, $W = \{1, .., w\}$, who return the judgment $R_z^{(y)}$ about the correctness of the response z. $R_z^{(y)}$ is either 0 if the reviewer states that the response is correct or 1 if he states that it was incorrect. The consolidated estimate for the actual quality of response z is denoted with $T_z \epsilon \{0; 1\}$ and the estimate for the fraction defective of the sample with $\widehat{p}$.

When leveraging reviewers out of the crowd and taking the service requesters' quality expectations as a benchmark it is obvious that the reviewers may perform errors. According to the definition in section 4.3.2, two types of errors can occur: A type I error happens if a reviewer considers a response incorrect even though it is actually correct. Respectively, a type II error happens if the reviewer considers an incorrect response correct. Corresponding error rates are introduced that represent the portion of type I and type II errors performed by each reviewer. A reviewer's type I error is denoted as $e_1^{(y)}$ and a type II error respectively as $e_2^{(y)}$. The challenge is to estimate the error rates $e_1^{(y)}$ and $e_2^{(y)}$ as well as the task quality T_z and the fraction defective $\widehat{p}$ in each sample.

To summarize, the given parameters are:

- $S = \{1, .., s\}$ is the sample of responses to be rated.
- r is the number of reviews per task.
- $W = \{1, .., w\}$ is a set of reviewers.
- $R_z^{(y)}$ is the judgment of reviewer y regarding the quality of task z. As not all reviewers are involved in the reviews for all tasks, $R_z^{(y)}$ is only defined for $z \in K_y$ with

$$K_y = \{z \mid \text{reviewer y gave a judgment on the result of task } z\} \ . \tag{6.27}$$

The following parameters are to be determined by leveraging the maximum likelihood method:

- $e_1^{(y)}$ is the estimated type I error rate of reviewer y.
- $e_2^{(y)}$ is the estimated type II error rate of reviewer y.
- $T_z \epsilon \{0; 1\}$ is the estimate for the true result rating regarding task z.
- $\widehat{p}$ is the estimated fraction defective.

Applying the maximum likelihood method

The primary challenge to be addressed by the maximum likelihood method is to find estimates for the type I and type II error rates and for the fraction defective of the responses which are good enough to meet the quality requirements of the requester. The W observers in Dawid & Skene's approach correspond to the reviewers. The v categories correspond to two possible ratings regarding the response quality: correct or incorrect. The items z correspond to the sample S of responses to be rated. The number of times observer y states that subject z belongs to category x, namely $n_{zx}^{(y)}$, corresponds to judgments of the reviewers $R_z^{(y)}$.

A difference here is that in the group validation model, the values are binary and not enumerative. Thus, if reviewer y thinks that task z belongs to category x, then $n_{zx}^{(y)} = 1$, otherwise $n_{zx}^{(y)} = 0$. As there are only two categories, $n_{z0}^{(y)}$ and $n_{z1}^{(y)}$ always sum up to 1 because one of the categories has to be chosen. Therefore, only $R_z^{(y)} := n_{z1}^{(y)}$ needs to be defined. As not all reviewers judge each task, $R_z^{(y)}$ is only defined if reviewer y gives a judgment about the quality of response z.

It remains to apply the concept of the probability that subject z belongs to class x which was denoted as T_{zx}. In this model T_{z1} is the probability that the response is correct, $P(R_z^{(y)} = 1)$, and T_{z0} the probability that it is not, $P(R_z^{(y)} = 0)$. Since $T_{z1} = 1 - T_{z0}$, only one value $T_z = T_{z1}$ is needed.

The estimated fraction defective is the probability of category "incorrect" ($x = 1$). It can be calculated with

$$\widehat{p} = \sum_{z \in S} \frac{T_z}{|S|} \, . \tag{6.28}$$

Accordingly, the probability of category "correct" is

$$1 - \widehat{p} = \sum_{z \in S} \frac{1 - T_z}{|S|} \, . \tag{6.29}$$

For all reviewers that have been involved in at least one review, the type I error is calculated with

$$e_1^{(y)} = \pi_{01}^{(y)} = \frac{\sum\limits_{z \in K_y} (1 - T_z) \cdot R_z^{(y)}}{\sum\limits_{z \in K_y} (1 - T_z) \cdot (1 - R_z^{(y)}) + \sum\limits_{z \in K_y} (1 - T_z) \cdot R_z^{(y)}} \,. \tag{6.30}$$

Respectively reviewer y's type II error rate is obtained with

$$e_2^{(y)} = \pi_{10}^{(y)} = \frac{\sum\limits_{z \in K_y} T_z \cdot R_z^{(y)}}{\sum\limits_{z \in K_y} T_z \cdot (1 - R_z^{(y)}) + \sum\limits_{z \in K_y} T_z \cdot R_z^{(y)}} \,. \tag{6.31}$$

The calculation of $\pi_{00}^{(y)}$ and $\pi_{11}^{(y)}$ as reviewer y's success rates is performed analogously.

The posterior probabilities of an error with regard to task z can be calculated as

$$P(T_z = 1 \mid \text{data}) = P(\text{data} \mid T_z = 1) \cdot P(T_z = 1) \,. \tag{6.32}$$

what in numerical terms is:

$$P(T_z = 1 \mid \text{data}) = \frac{\prod\limits_{y \in W} (\pi_{10}^{(y)})^{1-R_z^{(y)}} \cdot (\pi_{11}^{(y)})^{R_z^{(y)}} \cdot \widehat{p}}{\prod\limits_{y \in W} (\pi_{00}^{(y)})^{1-R_z^{(y)}} \cdot (\pi_{01}^{(y)})^{R_z{}^{(y)}} \cdot (1 - \widehat{p}) + \prod\limits_{y \in W} (\pi_{10}^{(y)})^{1-R_z^{(y)}} \cdot (\pi_{11}^{(y)})^{R_z{}^{(y)}} \cdot \widehat{p}} \tag{6.33}$$

The iterative procedure remains unchanged.

Unless estimates are available from the inspection of previous samples, the averages of the reviewer judgments $R_z^{(y)}$ are used as initial estimates for the probabilities T_z:

$$\widehat{T}_z = \frac{\sum\limits_{z \in K_y} R_z^{(y)}}{r} \tag{6.34}$$

6.2.6. Number of reviewers to be used

The Dodge-Romig acceptance sampling plan assumes that sample inspection is perfect. However, when using the maximum likelihood estimation for sample inspection, there is no hint about the statistical significance of the inspection decisions. It just returns the most likely configuration.

It is obvious that the significance is increasing with a growing number of reviewers. From an economic perspective however, the number of reviewers should be minimized in order to make the approach efficient. This section aims to provide a tradeoff decision by estimating the minimal number of reviewers needed while reaching a specific significance of the review decision.

The approach does not take the maximum likelihood estimation into consideration but uses a simplified inspection mechanism as a model that is based on an analogy to acceptance sampling. The basic objective is to determine how many reviewers need to be asked and how many of them need to accept or reject the worker's response in order to come to a reliable joint review decision. For simplification it is assumed that there is a large number of reviewers available to whom the same type I and type II errors e_1 and e_2 can be assigned. The analogy to acceptance sampling is described in the following:

- The lot is represented by a hypothetical pool of review decisions performed by a large number of reviewers, each item representing the decision of a single reviewer. A good item means that the reviewer accepts the worker's response and a bad item means that the reviewer rejects the response.
- The actual quality level (fraction defective) of the lot is represented by the average review decision of all reviewers. Because the same type I and type II errors are assumed for all workers, this average is either approximately e_1 or approximately $(1 - e_2)$, depending on whether the response to be reviewed is actually correct or not. If it is correct, an average of e_1 workers will state that it was not (type I error). If the response is incorrect, an average of $(1 - e_2)$ workers will state that it was actually correct (type II error).
- The sample is represented by the review responses of a number of n concrete reviewers who each decide whether to accept or to reject the response.

The objective is to accept (or reject) the worker's response if there is a high probability $\beta = (1 - \alpha)$ that the sample of reviewer decisions reflects the opinion of the hypothetical pool of reviewers. The idea is basically to ask enough reviewers to find out with sufficient significance $\beta = (1 - \alpha)$, whether the average reviewer decision is to accept or to reject the response.

The required sample size n (number if reviewers) can be determined by the Guenther algorithm described in figure 4.5 of section 4.3.2. The algorithm can be used to design a sampling plan for the problem by specifying the parameters AQL, RQL and the significance levels α and β. AQL and RQL are associated with type I and type II error rates e_1 and e_2, $\beta = (1 - \alpha)$ is associated with the desired significance of the actual joint reviewer decision.

The error rates e_1 and e_2 are determined by an initial run of the EM algorithm:

$$e_1 = \frac{1}{r \cdot n} \cdot \sum_{y \in W} e_1^{(y)} \cdot l^{(y)} \tag{6.35}$$

$$e_2 = \frac{1}{r \cdot n} \cdot \sum_{y \in W} e_2^{(y)} \cdot l^{(y)} \tag{6.36}$$

The following example illustrates the considerations: Calculating an example plan with the Guenther algorithm for the parameters $e_1 = 0.1$, $e_2 = 0.2$, and $\beta = 1 - \alpha = 0.05$ results in a sample size $n = 6$ and acceptance number $c = 2$. This means, if the average type I and type II error rates of the reviewers can be expected to be 0.1 and 0.2, a group of 6 reviewers needs to be asked in order to decide with a significance of 0.05 whether to accept or to reject a worker response.

The acceptance number c which is calculated along with the sample size is only needed when using the sampling plan as an alternative to the MLE. Then, the process would be the following: For each response to be validated, n reviewers would be asked to accept or reject it. If less than c rejected it, it would be still accepted, otherwise it would be actually rejected.

However, if pursuing that direction, the sequential sampling plan mentioned in section 4.3.2 should be considered as an alternative in order to increase the efficiency.

Part III.

Evaluation

7. Toolkit development

This chapter introduces a software toolkit that has been developed within the scope of this thesis as a technical foundation for evaluating the QM mechanisms described in the previous chapters.[1] The toolkit allows for performing experiments on top of commercial cloud labor platforms, either based on live experiments or simulations. The toolkit specifically supports the CSP/DVM model introduced in chapter 5 as well as the DVM for multi-labelling introduced in chapter 6. The chapter is structured as follows: Section 7.1 describes the toolkit architecture. Afterwards, section 7.2 explains how real-time experiments can be conducted with the tool whereas section 7.3 addresses the concept of the simulation mode.

7.1. Architecture

From a software engineering point of view, the toolkit can be utilized to implement a variety of usage scenarios. For instance, the toolkit might act as a software framework to implement third party quality management services that mediate between requesters and a cloud labor platform. It could also be used to complement existing client components with QM capabilities. Finally, the toolkit can directly be used as a software client for commercial cloud labor service platforms. A browser based progress monitor allows for tracking the execution progress during task execution. The built-in simulation platform allows for comparing the performance of different QM approaches or different configurations of the same QM approach.

Figure 7.1 illustrates the general architecture of the toolkit, which comprises the following core services:

- The *configuration manager* governs the different cloud labor scenarios to be performed by the toolkit. All scenario specific configurations, sources and results are wrapped into separate resource pools, so switching between different scenarios or different versions of the same scenario is seamlessly possible.

[1] Considerable parts of this chapter have already been published at the 3rd Human Computation Workshop of the 25th Conference on Artificial Intelligence (Bermbach et al., 2011).

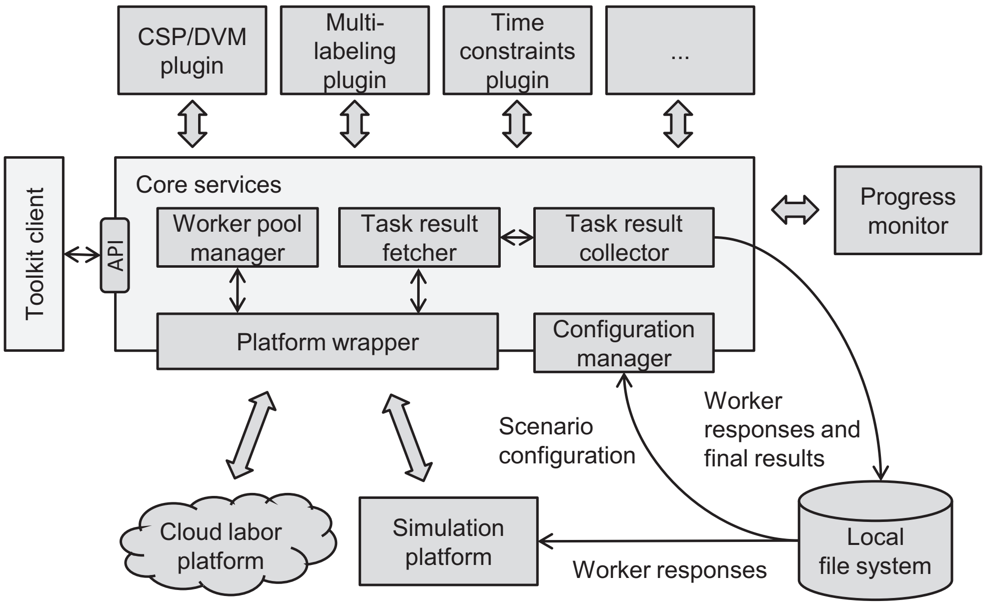


Figure 7.1.: Architectural overview of the CSP/DVM toolkit.

- The *task result collector* stores worker submissions internally and persists them on the file system. This component also groups submissions for assignments of the same task. The individual plugins may store worker submissions or retrieve them as complete sets.
- The *task result fetcher* periodically polls the cloud labor platform for worker submissions and ensures that a single submission is only processed once. Furthermore, it provides an internal queuing system to the plugins.
- The *platform wrapper* represents the interface to commercial cloud labor platforms as well as to the internal simulation platform. So far, a wrapper for the MTurk platform is available that uses its SOAP-based Web service interface. The toolkit can be easily extended with additional platform wrappers.
- The *worker pool manager* is responsible for registering qualification tests and assessing candidates on the fly. Workers are then assigned to different worker pools in order to control which worker may access certain tasks and which may not. This component also provides import and export functionality for worker data and triggers synchronization with the underlying cloud labor platform.

Individual QM features are implemented as plugins. Within the scope of the thesis, plugins for accuracy management and time constraints have been developed:

- The *CSP/DVM* plugin implements the CSP/DVM model. It monitors incoming responses and dynamically decides whether additional assignments for a task need to be published before coming to a conclusion. Finally, the result is returned to the result collector or the task is escalated. Custom accuracy management plugins may be developed that process incoming responses in a different way.
- The *multi-labeling* plugin implements the DVM for multi-labeling developed in section 6.1. It provides features for *merging* responses from multiple workers, i.e. to assemble a final result from labels returned by different workers.

- The *time constraints plugin* dynamically adjusts the size of the worker pool according to the considerations in section 5.4.3. The current capabilities are only rudimentary and could be used as a starting point for further developments.

The toolkit was developed using Java SE 6 and is currently maintained as an Eclipse 3.6 project. All components described above are either implemented as separate classes or as a set of classes. Extensibility is generally achieved through usage of abstract classes and interfaces combined with reflections. Regarding the physical design, the toolkit is primarily realized as a library which can be used for multiple purposes, e.g. within a Web application or a standalone tool. Initializer classes can easily be added by adapting the existing standalone initialization class. The current implementation focuses on the MTurk platform and, hence, includes an implementation of the platform wrapper component for this provider. The toolkit is available as open source under MIT license (Kern et al., 2012a). A user's guide provided with the source code explains how to customize and use the tool. The following two sections illustrate the two general modes of operation, the *live* and the *simulation* mode.

7.2. Live mode

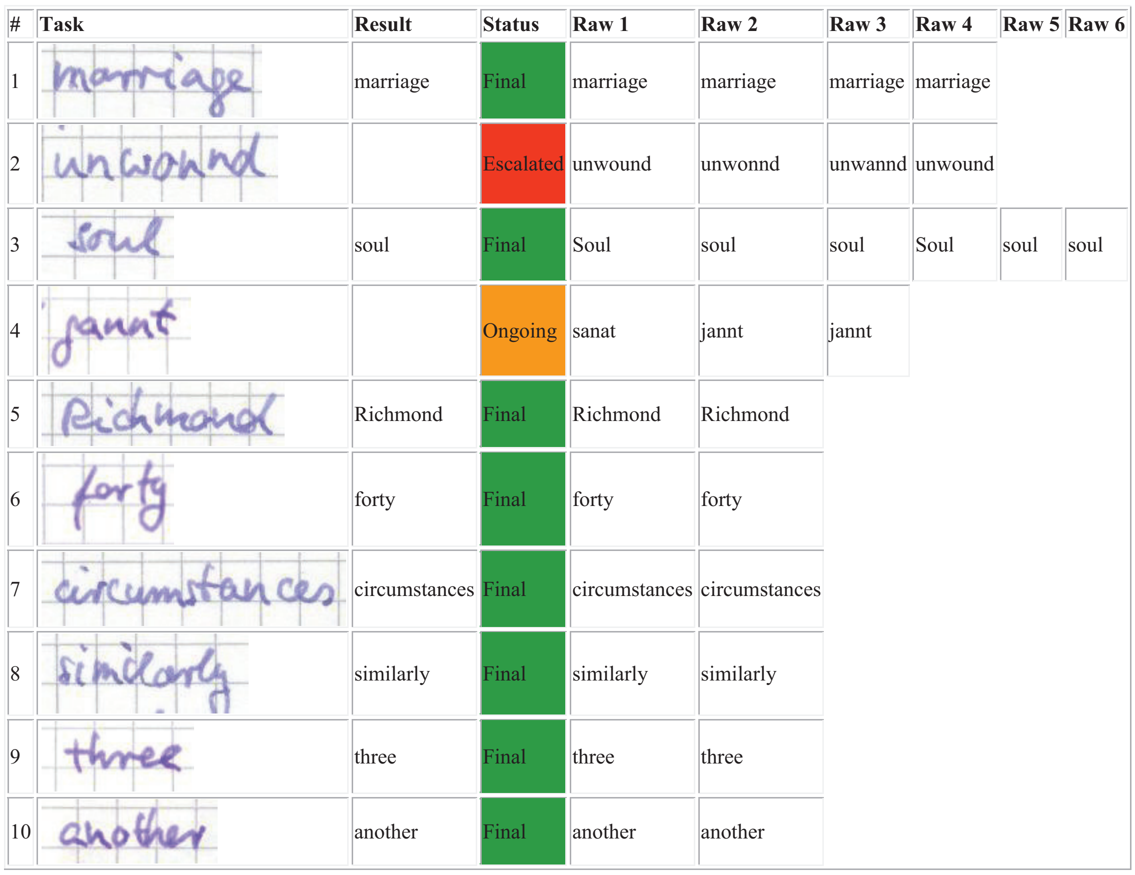
In the live mode, the user may provide a batch of tasks which are published to a commercial cloud labor platform where they can be picked up and completed by real workers. The responses are dynamically fetched from the platform and evaluated by the correctness plugin. According to the configuration of the QM plugin, additional task assignments may be automatically published as needed.

A certification test can be associated to the task type that needs to be successfully completed by new workers before they will be allowed to work on tasks. The CSP/DVM toolkit will automatically assess the responses in real-time by comparing them with the gold standard provided for the test. Only such workers who provide correct answers to a certain minimum number of the test questions will be allowed to contribute.

A browser based progress monitor allows for tracking the actual task execution. The screen is automatically updated in regular time intervals. Figure 7.2 shows an example screenshot of the process monitor from an experiment with 10 handwritten words. Column "Task" displays the words to be recognized while columns "Raw 1" to "Raw 6" represent the individual responses from different workers which are finally consolidated by the QM plugin in column "Result". Column "Status" shows that for most of the words, a final result could be identified while one task (task 2) had been escalated back to the requester. Task 4 had still been in progress when the screenshot was taken. The experiment has been performed in *full inspection*[2] mode. One of the first responses of task 1 contained a leading space which is invisible on the screenshot. However, even additional spaces have been interpreted as discrepancies in this example, thus additional responses had been requested. Task 3 indicates that capitalization also matters. All the results identified by the DVM adhere to the gold standard.

The live mode has two drawbacks. First, performing experiments can be expensive and time consuming. Especially when investigating the effect of configuration parameters like $AOQL$, the clearance number i and the sample fraction f, an extensive number of experiments may be needed. Even for simple tasks, this could quickly generate costs in

[2] Refer to section 5.6.2.



#	Task	Result	Status	Raw 1	Raw 2	Raw 3	Raw 4	Raw 5	Raw 6
1		marriage	Final	marriage	marriage	marriage	marriage		
2			Escalated	unwound	unwonnd	unwannd	unwound		
3		soul	Final	Soul	soul	soul	Soul	soul	soul
4			Ongoing	sanat	jannt	jannt			
5		Richmond	Final	Richmond	Richmond				
6		forty	Final	forty	forty				
7		circumstances	Final	circumstances	circumstances				
8		similarly	Final	similarly	similarly				
9		three	Final	three	three				
10		another	Final	another	another				

Figure 7.2.: Example process monitor output of the CSP/DVM toolkit.

the amount of thousands of dollars. Second, it can be challenging to perform a series of tests under comparable conditions, because the actual availability of workers varies. The simulation mode described in the following section is intended to address the two issues.

7.3. Simulation mode

Compared to the live mode, the simulation mode drastically reduces the cost and the duration of test series because workforce costs are only incurred once when worker responses are initially recorded. The pre-recorded responses can then be used again and again for various simulations. The simulated tests will also allow for controlled experiments because multiple experiments can be performed based on the same contributions of the same workers in order to balance the impact of worker availability.

The simulation mode differentiates between two phases: The *input phase* and the *execution phase*. In the input phase, a set of responses is gathered from the cloud labor platform which is then used in the execution phase as a basis for the actual simulation of the QM approach.

For the *input phase*, a batch of tasks is processed by passing a well defined number of assignments of each task to the cloud labor platform. For a given task, each assignment must be performed by a separate worker. The incoming responses are stored in an internal data store on the local file system. Like in the live mode, a certification requirement can be applied, which must be met by new workers before they are allowed to contribute. The test responses will be automatically assessed by the tool in real-time.

In the *execution phase*, the pre-recorded responses are retrieved by the simulation platform as they are needed. The platform wrapper provides the exact same interfaces for the simulation platform as for a real cloud labor platform. There are functions to create tasks as well as task assignments and to retrieve the incoming responses. However, as opposed to a real cloud labor platform, the responses are not generated by the workers in real-time but they are taken from the internal pool of available responses that had been acquired during the input phase. Nevertheless, they are *real* responses that have been provided by *real* workers. For each simulation run, only the required number of responses is retrieved from the pool, i.e. not necessarily all of the prerecorded responses are used. That way, the dynamic nature of the DVM is simulated. Depending on the configuration of the QM plugin, more assignments may be needed for the simulation than had been pre-recorded previously. Such situations are logged and reported in the output of the tool. In order to support a realistic simulation, the tasks and also the task assignments are being returned by the simulation platform in random order.

In chapters 8 and 9, the two modes of the toolkit will be leveraged for assessing the CSP/DVM in a model scenario and in two case studies.

8. Evaluation of core model

The evaluation of the core model has been performed in two complementary ways. First, it has been applied to a model scenario in order to validate its effectiveness and efficiency. Second, the applicability of the model has been proven in a business scenario in cooperation with industry partners.

Section 8.1 introduces the model scenario and describes the results of the corresponding experiments. Section 8.2 then covers the business scenario which represents the first of three case studies being discussed in this thesis.

8.1. Model scenario: Optical character recognition

Data revision represents a class of applications that is widely used on commercial cloud labor platforms like MTurk. It includes applications such as classifying, tagging, summarizing and revising content, and transcribing audio and video data. A specific form or data revision is the recognition of written texts, also called *optical character recognition* (OCR). Like many data revision tasks, OCR of handwritten texts cannot be fully automated yet (Lopresti, 2009). Even sophisticated technologies need human assistance in order to achieve satisfactory results. OCR has been chosen as a model scenario mainly because of three reasons. First, because it represents a realistic use case. Second, because it does not require specific skills. And thirdly, because only little effort is needed for recognizing words. Thus, the duration and the costs of experiments can be kept low.

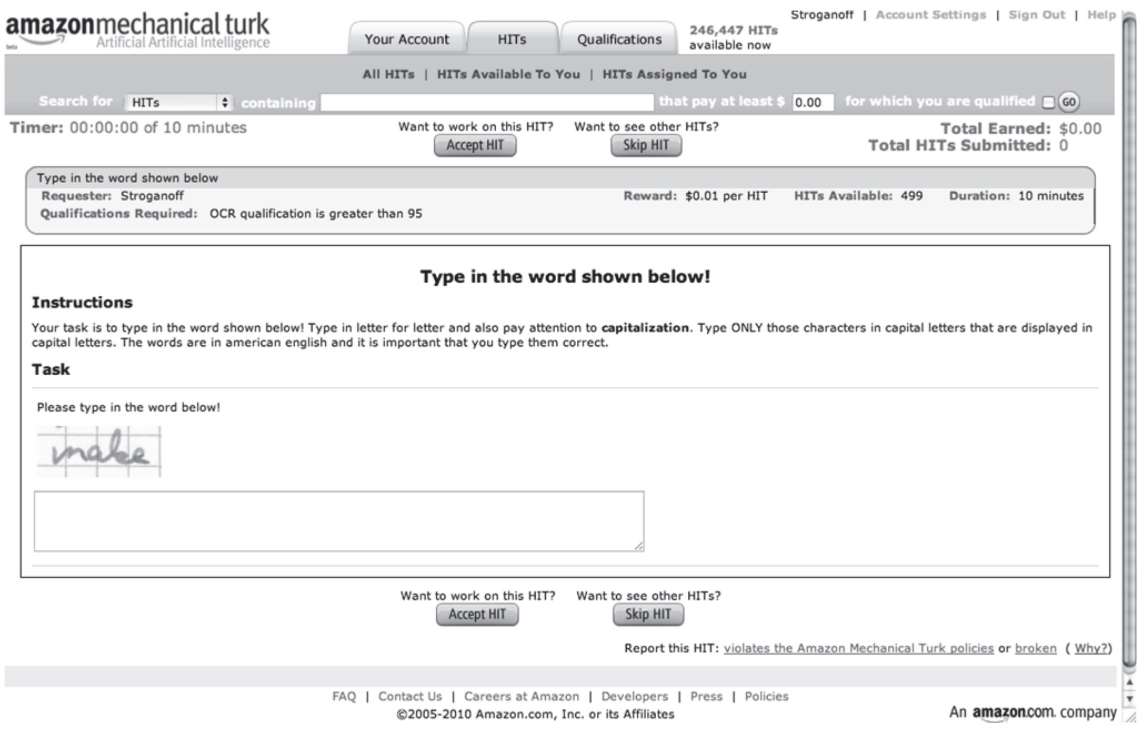
The structure of this section[1] is as follows: After introducing the experimental setup in section 8.1.1, the results of the experiments are divided into two parts. Section 8.1.2 elaborates on the simulations while section 8.1.3 elaborates on the real-time experiments. Sections 8.1.4 and 8.1.5 then compare and discuss the results of the two types of experiments.

8.1.1. Experimental setup

The experiments have been performed on top of MTurk with the help of the CSP/DVM toolkit presented in chapter 7. The simulation mode of the tool was used to simulate

[1] Considerable parts of this chapter have already been published in the International Journal of Cooperative Information Systems (Kern et al., 2012b).

experiments based on pre-recorded worker responses while the live mode was used to conduct real-time experiments. For both types of experiments, the same data set consisting of 1176 handwritten words was used. In each of the tasks, a worker was asked to read and type in a single handwritten word which was displayed as an image file (JPEG) in the task interface. The gold standard was specified by the author of the handwriting himself as he should know best what his writing is supposed to mean. In order to increase the difficulty of the tasks, capitalization had to be considered by the workers and no additional characters (e.g. spaces or newline characters) were supposed to be entered. Figure 8.1 provides a screenshot of the OCR task interface on the MTurk platform. The task instructions and the image file are shown in the lower half of the screen. In the rounded box above, the task parameters like requester name, qualification requirements, reward per task[2], number of available tasks and maximum working time are presented. Workers earned a wage of \$0.01 per completed task. The maximum time allotted to work on a task (duration) was 10 minutes, even though a task can usually be completed in a few seconds. Using the button "Accept HIT" workers can claim a task in order to start working on it, with the "Skip HIT" button they can start over and jump to the next task.


amazonmechanical turk
Artificial Artificial Intelligence
Stroganoff | Account Settings | Sign Out | Help
Your Account
HITs
Qualifications
246,447 HITs available now
All HITs | HITs Available To You | HITs Assigned To You
Search for HITs containing that pay at least $ 0.00 for which you are qualified GO
Accept HIT
Skip HIT
FAQ | Contact Us | Careers at Amazon | Developers | Press | Policies
An amazon.com company

Timer: 00:00:00 of 10 minutes

Want to work on this HIT? Want to see other HITs?

Total Earned: \$0.00
Total HITs Submitted: 0

Type in the word shown below
Requester: Stroganoff — Reward: \$0.01 per HIT — HITs Available: 499 — Duration: 10 minutes
Qualifications Required: OCR qualification is greater than 95

Type in the word shown below!

Instructions

Your task is to type in the word shown below! Type in letter for letter and also pay attention to **capitalization**. Type ONLY those characters in capital letters that are displayed in capital letters. The words are in american english and it is important that you type them correct.

Task

Please type in the word below!

Want to work on this HIT? Want to see other HITs?

Report this HIT: violates the Amazon Mechanical Turk policies or broken (Why?)

©2005-2010 Amazon.com, Inc. or its Affiliates

Figure 8.1.: Worker interface of an OCR task on the MTurk platform.

All workers had to pass an unpaid qualification test before being allowed to participate in the actual experiment. The test consisted of a series of 10 OCR tasks (i.e. 10 words). Nine of the words had to be typed in correctly in order to achieve a failure rate of $p = 0.1$.

8.1.2. Simulations

As a basis for the simulations, a batch of 11,760 assignments (10 assignments for each of the 1,176 tasks) was uploaded to the MTurk platform on February 1st, 2010. A single

[2] Amazon uses the term *human intelligence task* (HIT).

worker was not allowed to handle more than one assignment of the same task. Given a payment of \$0.01 per task, a service charge of \$0.005 per task and a total number of $1,176 \times 10 = 11,760$ assignments, the total expenses for the experiment summed up to $11,760 \times$ (\$0.01+\$0.005) =\$176.40.

Execution performance

One of the first visible and most astonishing results of the experiment was certainly the speed at which the worker responses were submitted. In the first pretests, a batch of 3,528 tasks was completed by 112 workers in less than 15 minutes at an execution rate of 14,088 tasks per hour. The first response arrived after 22 seconds. During other experiments a total execution speed of up to 3 times as fast had been observed because more workers participated. It can be assumed that the execution speed depends on the time of day, since most workers are located in the U.S. or in India (Ross et al., 2010).

Figure 8.2 illustrates the processing of the tasks on the MTurk platform during the input phase of the simulation experiments: 11,760 tasks have been performed by 38 workers in about 2:40 hours. The horizontal axis represents the time, starting with 0 at the beginning of the experiment. The vertical axis shows a consecutive number for each worker that joins the experiment. Each dot stands for a single worker response arriving at the platform while each line of dots represents the contributions of a single worker. The first worker started

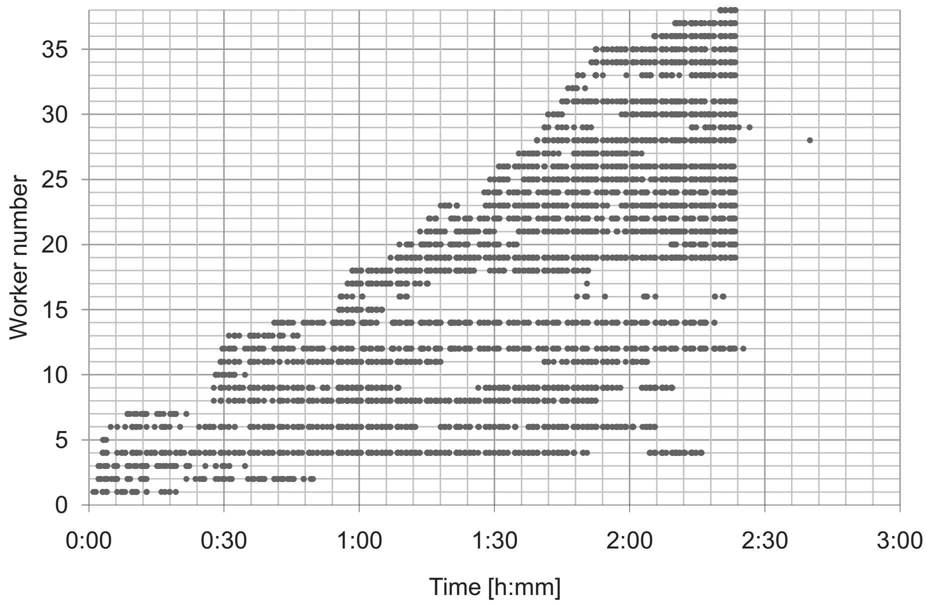


Figure 8.2.: Incoming worker responses when processing a batch of OCR tasks on MTurk.

after about 1 minute and stopped after about 20 minutes. Over time, more and more workers joined so that after a while, more than 20 workers have been working in parallel when suddenly, after 2:24 hours, there have been no tasks available any more. A small number of responses were delayed because they have been claimed and passed back after a while, when the workers realized that they cannot identify the handwritten word. After a series of workers had tried their luck, the final response eventually arrived after 2:40 hours.

Based on the 11,760 responses, two types of experiments have been performed: *Full inspection* and *random inspection* experiments. They are described in the following.

Full inspection

In the first simulation experiment, the efficiency of the DVM has been evaluated independently of the CSP-1. *Full inspection* is a term of SQC that indicates that all items of a sample are inspected. For the experiment this means that the DVM is issued for all tasks and the CSP-1 is actually not used at all. Instead of *full inspection* one could also say *DVM only*. However, in order to be consistent with the process flow of the overall model in figure 5.3, the term full inspection is preferred here. The process flow is still valid, but the CSP-1 is switched off by setting the sample fraction f to 1 or alternatively, by setting the clearance number i to a value higher than the number of tasks to be processed. In both cases, all responses submitted by the workers will be inspected.

Table 8.1 shows the parameters used for the full inspection experiment along with the results achieved. A quality goal of $\varphi_{min} = 0.95$ was applied along with an escalation limit of $\varepsilon_{min} = 0.01$. The initial number of possible responses was set to $v = 3$. This value had been determined in pre-experiments which indicated that for a given word, a group of workers would come up with an average of 3 distinct responses. If in a specific task, the actual number of distinct responses exceeded 3, v was increased dynamically. An accuracy of 98.3% was achieved with an average redundancy of 2.16. A fraction of 2.47% of the tasks were escalated, e.g. due to bad handwriting.

Table 8.1.: Results of the CSP/DVM simulation experiment in full inspection mode.

Parameters	Quality goal φ_{min}	0.950
	Escalation limit ε_{max}	0.010
	Response categories v	3
Results	Accuracy	98.3%
	AFI	100.0%
	Avg. redundancy	2.157
	Max. redundancy	4
	Escalated	2.5%

Random inspection with CSP-1

For the random inspection experiments, SQC was now enabled by switching on the CSP-1. In a series of tests, the CSP/DVM was evaluated for 3 different values of $AOQL$. Table 8.2 depicts the parameters along with the results. For $AOQL = 0.05$ (experiment 1), an average redundancy of 1.510 was observed, which is a significant improvement compared to the full inspection with a value of 2.16. This confirms that adding the CSP-1 actually reduces the QM effort and makes the approach more efficient. Only 41% of all tasks have been inspected and 1.8% of the tasks were escalated.

For $AOQL = 0.075$ (experiment 3) and even with $AOQL = 0.025$ (experiment 2), the quality objectives are again exceeded. However, there are situations where the model does not manage to achieve the desired level anymore. The reason for that lies in the gap between the gold standard and the majority decision of the workers: in several cases, the majority of the workers identified a certain word (e.g. ”five”) even if the writer (who represented the gold standard) had written a different word (e.g. ”fine”).

Figure 8.3 illustrates the efficiency of the CSP/DVM compared to the traditional majority vote approach. The vertical axis represents the accuracy of the results while the horizontal

Table 8.2.: Results of the CSP/DVM simulation experiments in random inspection mode.

Experiment		1	2	3
Parameters	$AOQL$	0.050	0.025	0.075
	Clearance number i	6	5	1
	Sample fraction f	0.249	0.582	0.036
	Quality goal φ_{min}	0.990	0.990	0.990
	Escalation limit ε_{max}	0.010	0.010	0.010
	Response categories v	3	3	3
Results	Accuracy	96.3%	98.2%	96.5%
	AFI	41.0%	66.4%	5.4%
	Avg. redundancy	1.510	1.800	1.250
	Max. redundancy	5	5	4
	Escalated	1.8%	2.6%	1.6%

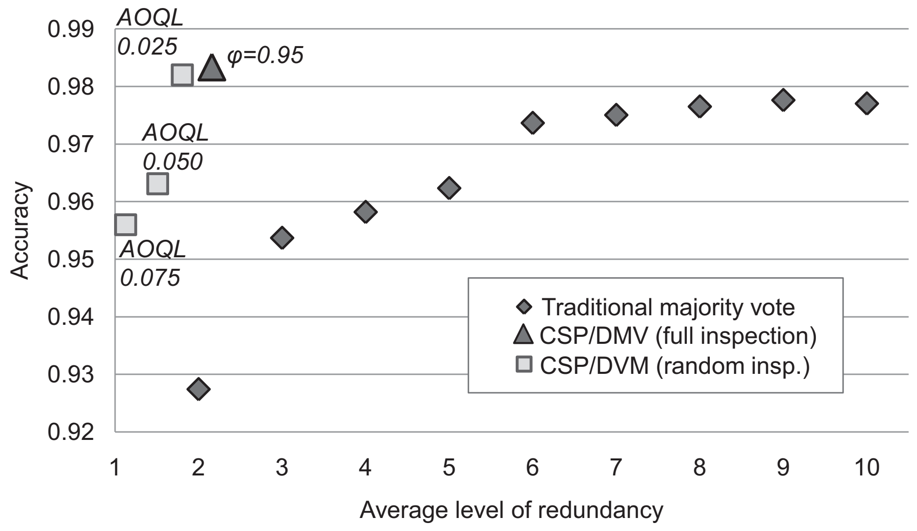


Figure 8.3.: Efficiency of the CSP/DVM approach in simulation mode compared to the traditional majority vote approach.

axis stands for the average redundancy, i.e. for the number of assignments per task needed by the DVM. The performance of the traditional majority vote is indicated by dark diamonds. It was simulated by drawing a fixed number out of the 10 assignments available in the data set for each task. For the two-fold majority vote, 2 assignments were used, three for the three-fold, and so on. Then, the response that occurred most frequently was chosen. If several responses occurred equally often (tie), a random choice between them was made, as suggested by (Snow et al., 2008). In order to balance statistical effects, the results of multiple drawings were averaged. In fact, all possible permutations were used. As an example, there are 10 ways to choose 9 of the 10 available assignments. The majority decision for each of them was calculated and the results were averaged in order to simulate the 9-fold majority vote.

The dark grey triangle indicates the result of the full inspection simulation experiment, in which the DVM even outperforms the accuracy of a ninefold traditional majority vote

(98.34% vs. 97.76%). This is a remarkable result given that the DVM is 4 times more efficient as it requires only 2.16 workers per task compared to 9 workers per task for the basic ninefold majority vote approach. In other words: the DVM approach has reduced the quality management effort by 76% compared to the traditional majority vote. This result is even exceeded by the results of the random inspection simulation experiments, which are indicated by the light grey squares in the figure. At $AOQL = 0.025$, the CSP/DVM outperforms the traditional 9-fold majority vote by 80% by only requiring an average of 1.8 workers per task.

8.1.3. Live experiments

For the live experiments, the same tasks were used as for the simulations. However, according to the concept of the CSP/DVM, initially only one assignment was published for each task. Depending on the CSP-1 status, the worker's historical failure rates and the actual responses, the QM approach dynamically decided in real-time whether additional assignments had to be published.

In September 2011, two live experiments have been performed on the MTurk platform using the complete data set of 1176 OCR tasks. For the first experiment, the parameters have been same as for simulation 1 in section 8.1.2, and for the second experiment same as in the simulation 2.

Execution performance

Figure 8.4 shows the execution performance of the full inspection live experiment, which took 47 minutes and to which 15 workers have contributed. For the first 27 minutes, the progress is similar as when gathering the responses for the simulations in figure 8.2. The

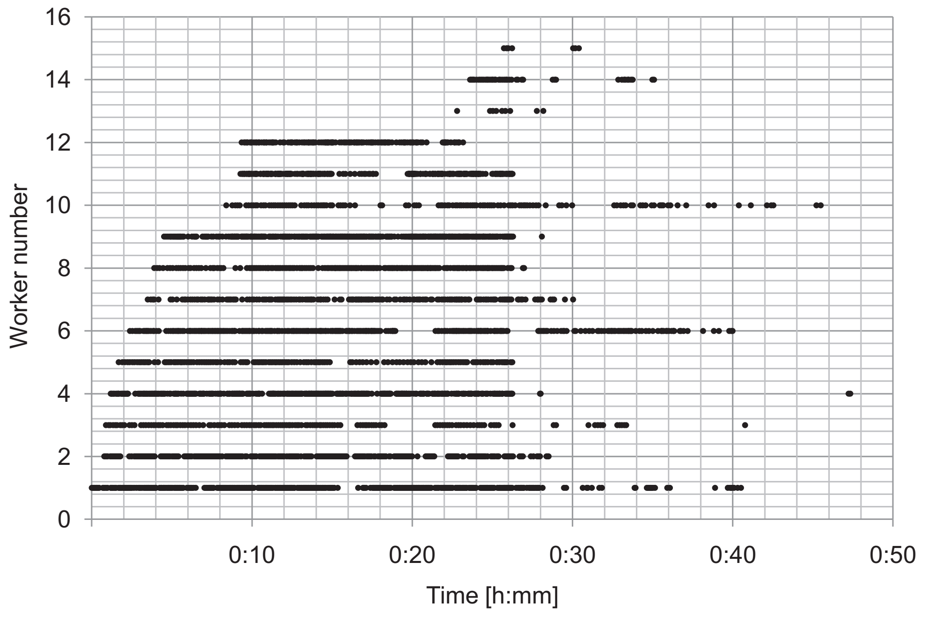


Figure 8.4.: Incoming worker responses when applying the CSP/DVM in full inspection mode to OCR tasks on MTurk.

number of participating workers grows even faster in the live experiment. However, after 27 minutes, several workers suddenly stop working or only sporadically submit responses.

This behavior can be explained by the dynamic behavior of the DVM. Towards the end, the continuous stream of tasks is interrupted because there are temporarily no assignments available any more. Therefore, the workers consider the process to be completed and go to find another task type to work on. However, as the DVM increases the redundancy sequentially, new assignments might be published dynamically depending on the responses provided for the previous ones. This applies in particular to complex tasks for which there is less agreement among the workers and thus a higher level of redundancy is required.

Figure 8.5 confirms this interpretation. The solid line represents the number of tasks being completed over time. Completion of a task means that all assignments of the tasks have been completed and based on these assignments, either a response has been returned to the requester or the task has been escalated. The dashed line represents the number of tasks that have been started, i.e. of which at least one assignment has been completed. According to the experimental setup, the first assignment of all tasks was made available

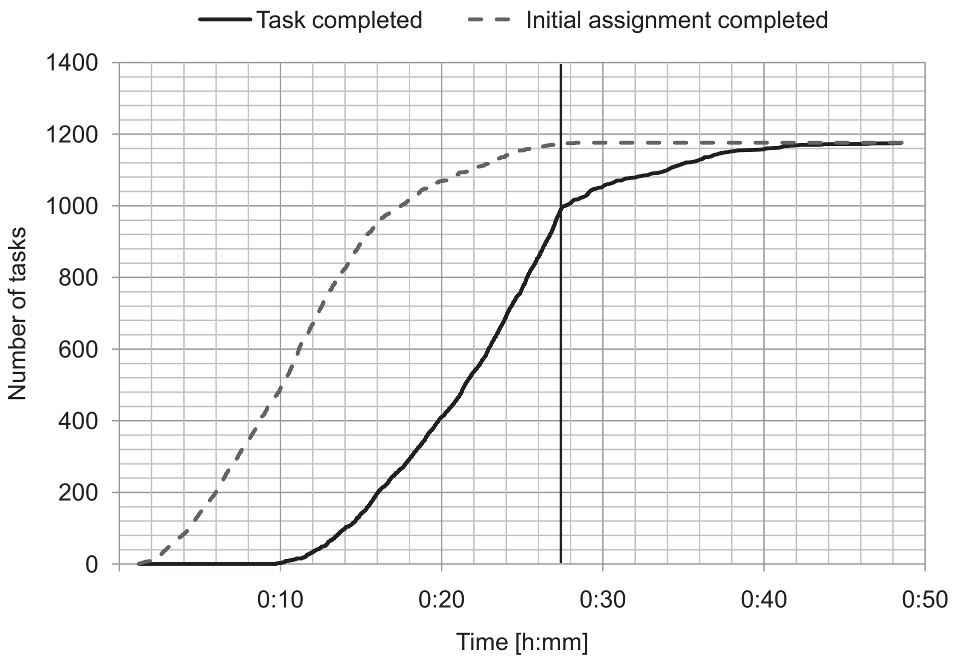


Figure 8.5.: Started and completed tasks during an experiment on MTurk.

at the beginning of the experiment. For the first 27 minutes, the task completion rate grows exponentially. Then, it suddenly slows down dramatically and slowly approaches the target of 1176 tasks. Comparing the two lines reveals that the solid line bends pretty much exactly when the initial assignments of all tasks have been completed. This point in time is indicated by a vertical line. From that moment on, there are obviously simply not enough new assignments generated by the DVM to keep all workers busy.

As illustrated by figure 8.6, the slowdown is even stronger in the random inspection experiment. The experiment took 2:52 hours and 13 workers participated. After 78 minutes, 1156 of the 1176 tasks had been completed. Then it took 94 more minutes to complete the remaining 20 tasks. The overall lower execution performance can be explained with the smaller number of contributing workers, which in return may be caused by the different time of the day the experiment was started.

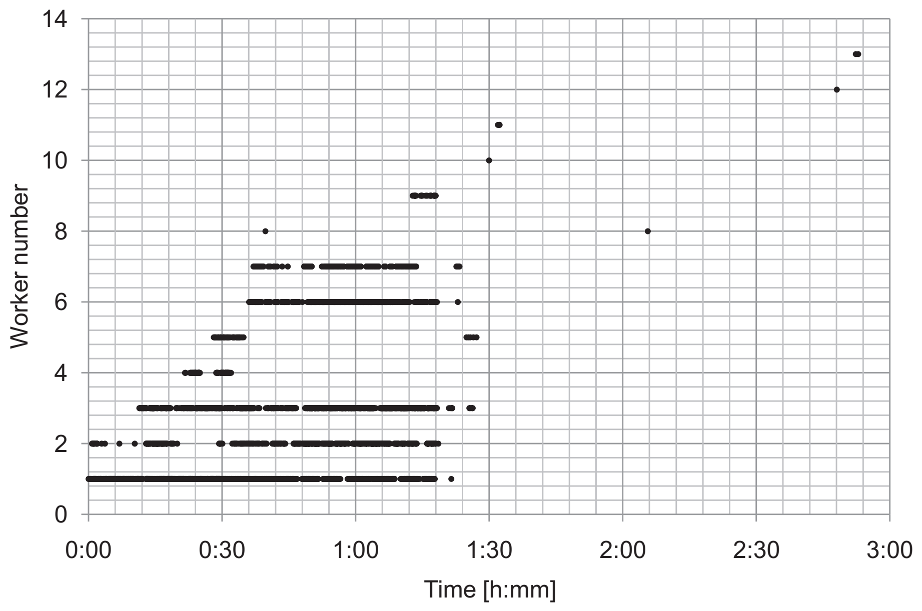


Figure 8.6.: Incoming worker responses when applying the CSP/DVM in random inspection mode to OCR tasks on MTurk.

Full inspection

Table 8.3 shows the parameters and the results of the full inspection live experiment. The quality level of 97.8% considerably overachieved the quality objective of 0.950. The redundancy had been 2.201 in average and 5 at maximum. A percentage of 3.7% of the tasks had been escalated.

Table 8.3.: Results of the CSP/DVM live experiment in full inspection mode.

Parameters	Sample fraction f	1.000
	Quality goal φ_{min}	0.950
	Escalation limit ε_{max}	0.010
	Response categories v	3
Results	Accuracy	97.8%
	AFI	100.0%
	Avg. redundancy	2.206
	Max. redundancy	5
	Escalated	3.7%

Random inspection with CSP-1

In the random inspection experiment depicted in table 8.4, the quality objective of 0.950 was again overachieved by reaching a quality level of 96.2%. With 1.734 assignments per task, the average redundancy had been considerably lower than in the full inspection mode. This again confirms that the CSP-1 has the capability of increasing the efficiency of the DVM. A percentage of 61.6% of the tasks have been inspected. The maximum redundancy has reached the same value as for the full inspection mode while with 8.0% of the tasks, the

Table 8.4.: Results of the CSP/DVM live experiment in random inspection mode.

Parameters	$AOQL$	0.05
	Clearance number i	6
	Sample fraction f	0.233
	Quality goal φ_{min}	0.990
	Escalation limit ε_{max}	0.010
	Response categories v	3
Results	Accuracy	96.2%
	AFI	61.6%
	Avg. redundancy	1.734
	Max. redundancy	5
	Escalated	8.0%

amount of escalated tasks had been more than twice as high. This effect will be discussed when comparing the results of the live experiments with those of the simulations in the subsequent section.

8.1.4. Comparison of simulation and live experiments

Comparing the execution performance of the data acquisition step in figure 8.2 to the one of the full inspection live experiment in figure 8.4 reveals that the execution progress can be similar in both situations as long as enough assignments are available to be claimed by the workers. In the live experiments, the execution progress significantly drops once there are temporarily no assignments available any more.

When comparing the results of the simulations to the live experiments in table 8.5, an important finding is that the actual quality objectives have been met in both types of experiments. Major differences concern the average redundancy, the AFI and the number of escalated tasks.

Table 8.5.: Performance of the CSP/DVM for different types of experiments (simulation versus live) and different quality objectives.

		Simulations				Live tests	
Parameters	Inspection	full	random	random	random	full	random
	$AOQL$	-	0.025	0.050	0.075	-	0.05
	Clearance number i	-	5	6	1	-	6
	Sample fraction f	1.000	0.582	0.233	0.036	1.000	0.233
	Quality goal φ_{min}	0.950	0.990	0.990	0.990	0.950	0.990
	Escalation limit ε_{max}	0.010	0.010	0.010	0.010	0.010	0.010
	Response categories v	3	3	3	3	3	3
Results	Accuracy	0.983	0.982	0.963	0.965	0.978	0.962
	AFI	1.000	0.664	0.410	0.054	1.000	0.616
	Avg. redundancy	2.157	1.800	1.510	1.250	2.206	1.734
	Max. redundancy	4	5	5	4	5	5
	Escalated tasks	29	30	21	19	44	94

According to the definition of the escalation limit, a task should be escalated if an exceptionally difficult task causes a disagreement among the workers, i.e. if the workers should have come to a higher level of agreement given their historical failure rates. However, knowing that the same tasks have been used for both experiments, the reason for the higher AFI in the live experiment must be caused by something else: According to equation (5.14), the value of the escalation limit solely depends on the responses and the estimated failure rates of the contributing workers. Thus, the higher AFI indicates that there had been a mismatch between the actual worker failure rates and their estimations calculated by the DVM. But how should the mismatch be introduced given that exactly the same parameters have been used for both experiments?

The estimations of the worker failure rates are updated each time when an inspection process is finished. Depending on the agreement among the workers, their failure rates are either increased or decreased. An important prerequisite for attaining good estimates is to have a good mix of difficult and simple tasks. If there is a majority of simple tasks, this will result in too optimistic estimations while a majority of difficult tasks will result in too pessimistic estimates. Both the simulation as well as the live experiment aim to achieve that requirement by processing the tasks in a random order. However, in the live experiments, workers have the (undesired) option to cherry-pick the simple tasks by skipping the difficult ones. Figure 8.7 compares the outcome of the CSP/DVM inspection processes for the simulated vs. the live experiments with an $AOQL$ of 0.05. The horizontal axis indicates the percentage of completed inspections. A percentage is used because the actual number of inspections differs in the two experiments according to the different values of the AFI. The vertical axis represents the percentage of inspections having resulted in disagreement. Disagreement means that at least one of the responses provided by the workers differs from the other ones. Only the final decisions of the DVM are considered, which result either in an escalation or in a final result being delivered to the requester.

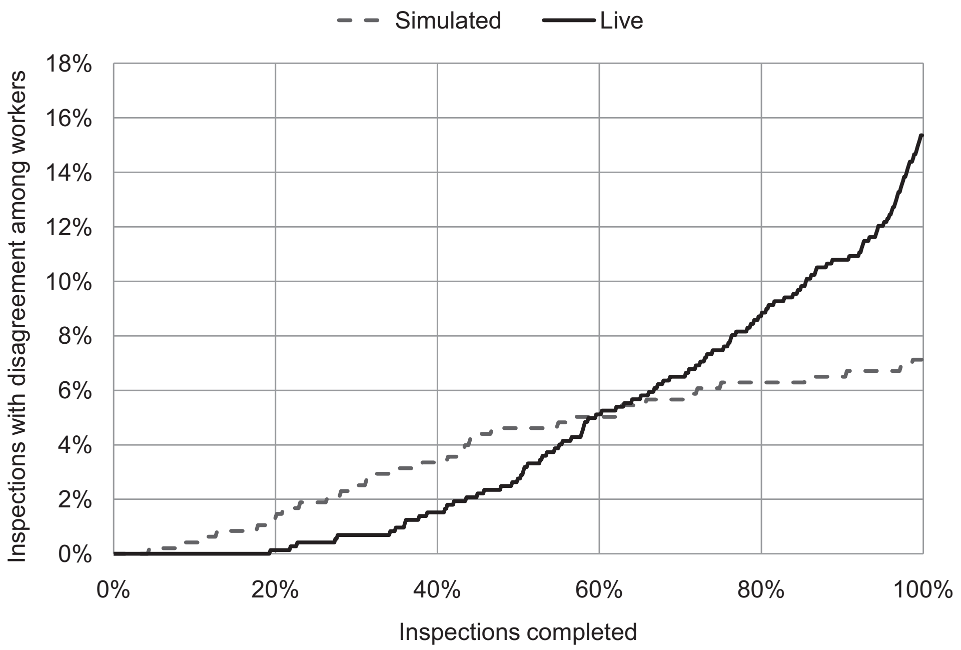


Figure 8.7.: Worker disagreement in the inspection process of the CSP/DVM.

In the early phase of the simulation experiment, a higher portion of inspections ended up in disagreement. Then, the rise of the disagreement slowly declines because more and more inspections can be finished with only two workers. This is because the failure rates of the good workers begin to distinguish from the ones of the bad workers, so two good workers are sufficient for successfully completing an inspection. In contrast, the curve of the live experiments is initially flat and then rises disproportionately. By skipping the difficult tasks, there is not any disagreement among the workers for the first 20% of the inspections resulting in equally low failure rates for all workers. Once the workers have to content themselves with the more difficult tasks, the disagreement rate increases. At this point in time, the estimated failure rates do not provide a realistic picture anymore and a large percentage of tasks is getting escalated. Due to the escalations, the failure rates are not updated, which even reinforces the effect. Thus, there is a high risk for the subsequent tasks to be escalated as well.

The high escalation rate also affects the average redundancy. While it has been comparable in the full inspection experiments (2.206 in the live experiment vs. 2.157 in the simulation), it differs by 14.8% when using the CSP-1 (1.734 vs. 1.510). This is not surprising because escalations typically result in a higher redundancy than successfully completed tasks. The increase of the *AFI* from 0.410 to 0.616 can be seen as another consequence of the cherry-picking. Because of the high escalation rate, many workers cannot reach the random inspection phase of the CSP-1.

8.1.5. Discussion

The experiments have demonstrated that for both the simulated and the live experiments the predefined quality objectives could be met. By dynamically adjusting to the actual quality of the incoming responses, the QM effort could be reduced by up to 80% compared to the traditional majority vote. The overall results are illustrated by figure 8.8.

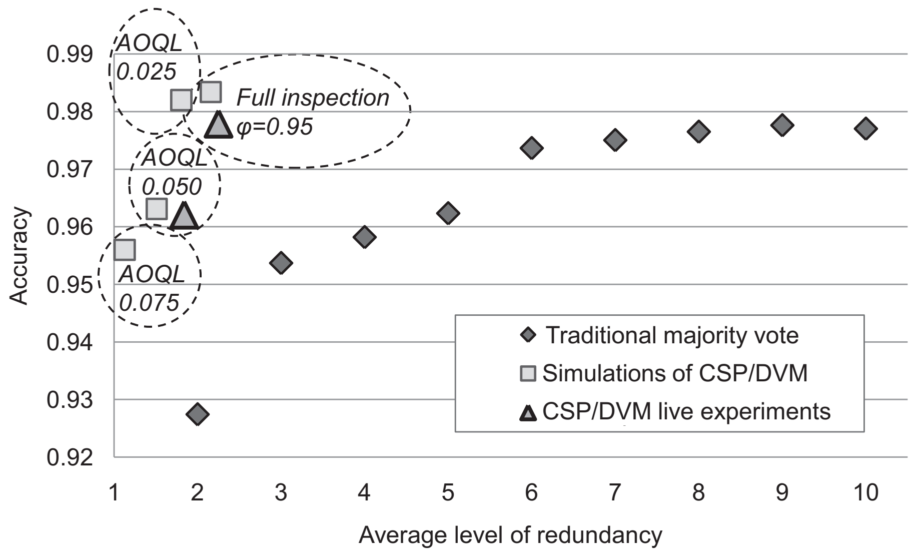


Figure 8.8.: Efficiency of the CSP/DVM approach compared to the traditional majority vote approach.

The execution performance of the experiments confirms the scalability of the CSP/DVM. From that point of view, it fulfills the general needs of an *efficient*, *scalable* and *goal-based* QM approach for cloud labor services as requested in section 3.4.2.

The experiments have also revealed a number of constraints that need to be considered when using the CSP/DVM. This refers to the decline of the execution progress after the initial assignments have been processed and the cherry-picking of simple tasks. First, there are several ways to avoid a discontinuity and speed up the finalization of the process: In full inspection mode, the execution process can be accelerated by leveraging the fact that always at least two assignments of each task need to be published. These assignments should be published immediately instead of waiting for the response of the first one before publishing the second one. In addition, for both the full and the random inspection, a discontinuity should be avoided by proactively publishing new assignments even before they are due. One option would be to switch the CSP/DVM to a fixed level of redundancy at the very end of the process. For example, the completion time for the live experiment 2 presented in figure 8.6 could have been split in half by turning the CSP/DVM switching to a higher redundancy for just the last 20 tasks.

Cherry-picking of simple tasks should ideally be completely avoided or at least reduced. Workers could still be given the option to skip a limited number of tasks but they should not have the freedom to pick and choose. While the built-in task interface of the MTurk allows for skipping tasks, a proprietary task interface may be plugged in by the requester in order to avoid that.

Even though OCR is a realistic use case for cloud labor services and it has been investigated on a commercial cloud labor platform with ”real” workers, it is still a model scenario that has been chosen merely for academic reasons. As a complement, the QM approach will be applied to a business scenario in the next section.

8.2. Case study: Address research

In the *address research* case study, the CSP/DVM approach has been tested under the conditions of a concrete business scenario in cooperation with two business partners. The scenario was provided by the German directory assistance provider *Telegate AG* and investigated on the commercial cloud labor platform *clickworker.com*.

The structure of the section is similar to the previous one: After introducing the address research scenario in section 8.2.1, section 8.2.2 illustrates the experimental setup. Then, sections 8.2.3 and 8.2.4 present the results of simulations and real-time experiments, which are eventually discussed in section 8.2.5.

8.2.1. Scenario

The cloud labor platform clickworker.com has already been introduced in section 2.5.1. It is operated by *Humangrid GmbH* in Essen, Germany, and went live in 2006. The platform has access to a workforce of about 300.000 workers and mainly focuses on text creation, search engine optimization, Web research, translation, tagging and categorization scenarios. Clickworker provides a growing number of self-service capabilities but also positions itself as a full-service crowdsourcing provider that is able to provide individual and customized solutions.[3]

The service cycle used by Clickworker is illustrated by figure 8.9. After an order has been placed by a requester along with the corresponding SLAs, projects are broken down into manageable tasks which are published to the workers. Once a worker has passed the qualification test being assigned to the specific type of task, he can begin working on the

[3] `http://www.clickworker.com/wp-content/uploads/2012/10/brochure_web_en.pdf`, last accessed on July 9, 2013.

tasks. In the quality assurance phase, the worker responses are validated in multiple ways, for example, texts created by workers are proofread and checked for plagiarism. The task results are then reassembled and made available for download to the requester.[4]

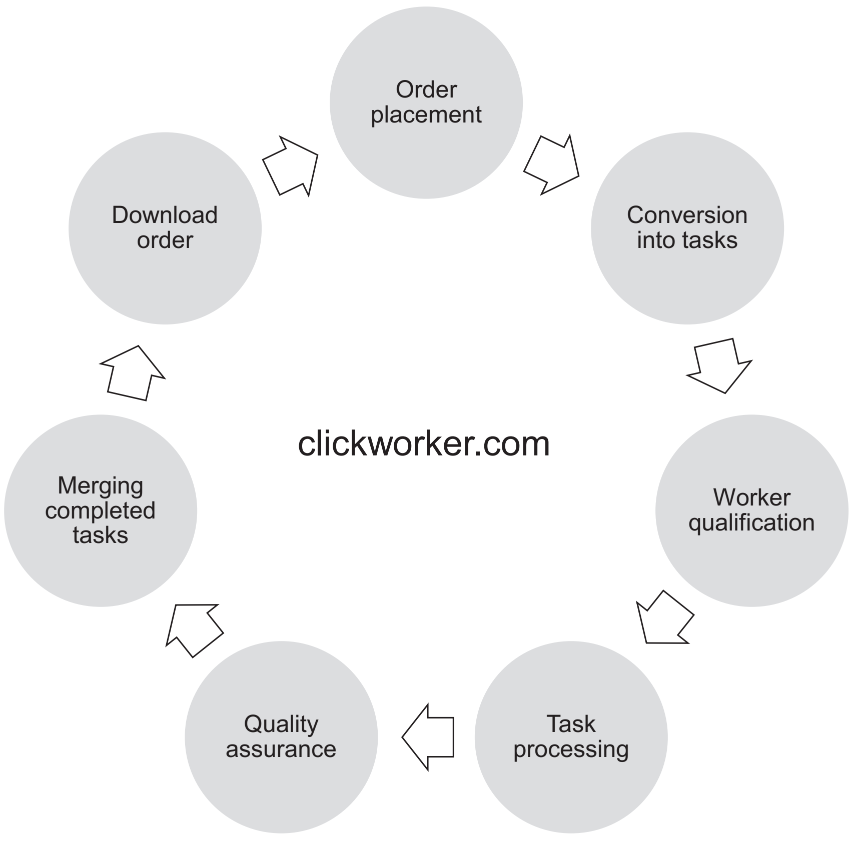


Figure 8.9.: Concept of the service provided by clickworker.com.

The scenario for this case study has been provided by *Telegate AG* which provides Web and phone based directory assistance across multiple channels. The company was founded in 1996 and has about 2,900 employees. In 2010, the annual revenue has been €124.6m.[5]

A key prerequisite for providing directory assistance is the availability of precise address and contact information of individuals and firms. A common application of cloud labor services is to research such address information from the Internet. The objective of the task outsourced to Clickworker has been to validate given addresses of restaurants, identify their URLs and their opening hours and classify them according to given categories. The specific research task performed in this case study is to identify the correct homepages of given restaurants. A subset of 1,000 addresses of restaurants had been provided for the experiment, consisting of the name, zip code, city, and street address. Most of the restaurants resided in hotels and many of them did not have a separate homepage. If they had, the link to that page was supposed to be retrieved rather than the one of the hotel. A unique ID was provided for each address in order to map the responses to the original requests.

[4] Derived from `http://www.clickworker.com/en/das-clickworker-prinzip/`, last accessed on July 9, 2013.

[5] `http://www.telegate.com/htm/en/Company/1039.htm`, last accessed on July 9, 2013.

The task instructions were designed in a way that allow for a unique identification of the homepage. Thus, the task can be seen as a deterministic task according to the definition in section 3.3.1. If the homepage could not be identified, the worker could indicate that by selecting an appropriate checkbox. Among the possible task responses, ”no url found” was handled as a separate unique response.

8.2.2. Experimental setup

On the clickworker.com platform, each project is represented by a so called *workflow controller* which is implemented using the open source Web development framework *Ruby on Rails*[6]. The individual tasks and assignments are supplied in form of *task input data items*, which are represented by XML documents. So called *nodes* define the actual task interfaces to be presented to the workers.[7]

For the case study, a new workflow controller was developed that implements the scenario as well as the CSP/DVM. The setup was used in two ways: First, for generating a batch of worker responses to be used for simulating the CSP/DVM using the toolkit introduced in chapter 7. Second, for conducting live tests by running the CSP/DVM directly on the clickworker.com platform.

Like other cloud labor platforms, the Clickworker platform provides a concept of qualification tests. As an entry criterion for contributing to the address research tasks, the workers had to have an *address research rating* of at least 75%. Given that this qualification type comprises not just URL research but also the identification of names, addresses, contact information and office opening hours, a success rate of 75% was assumed to correspond to a success rate of 80% for just the URL research task. Therefore, the failure rates of all contributing workers were initialized with a value of 0.2.

8.2.3. Simulation experiments

In November 2011, a number of 1001 URL research tasks was submitted to the Clickworker platform. For each of the tasks, 15 assignments were requested resulting in altogether 15,015 worker responses.

Gold standard

Telegate did have a large amount of data records available for the scenario but they did not know how reliable the data was. Therefore, it could not be used as a gold standard for the experiments. As discussed in chapter 3, defining a gold standard can be challenging. Even experts may come to different conclusions regarding what is the optimal response for a task.

The agreed approach for the case study was to generate a gold standard based on the feedback from multiple skilled workers. For each task, the 15 available responses were split into two partitions in the order they were submitted by the workers. The first 8 responses were used for generating the gold standard while the remaining 7 responses were used to perform the actual simulations. A response was accepted as the gold standard if it appeared at least 5 times among the first 8 responses. This had been the case for 904 of the 1001 tasks. For 97 tasks no gold standard could be defined. Nevertheless, the complete set of 1001 tasks was used for the simulations because it should be investigated how the CSP/DVM would handle the no-gold-standard tasks.

[6] `http://rubyonrails.org/`, last accessed on July 22, 2013.
[7] The platform is described in more detail in Meller (2012)

Execution performance

Figure 8.10 shows the execution performance when recording the 15,015 responses for the simulation experiments. Like in the equivalent charts in section 8.1, each dot stands for a response being submitted by a worker. The horizontal axis represents the submission time while the vertical axis represents the number of the worker in the order of their first submission. As opposed to the previous charts, zero on the time axis is not the start time of the experiment but midnight of day 1. The experiment was started at about 9 am and took about 2.5 days. Altogether, 409 workers have participated. Interestingly, there are two gaps during which there was almost no work performed at all. The first gap is between 12 and 15 hours from the starting point which corresponds to 12 p.m. and 3 p.m. on day 1. The second gap is between 24 and 36 hours which corresponds to midnight and noon on day 2. Presumably, the gaps occurred because workers went to lunch (day 1) or to bed (day 2), and they ended because the workers were informed by email that there are still tasks available. This explains the sudden worker increase after 15 and 36 hours. On day 3 there is also a period of reduced worker activity between midnight and 6 a.m. This maps to nighttime in Germany and indicates that the scenario was limited to German speakers, who are primarily located in Germany.

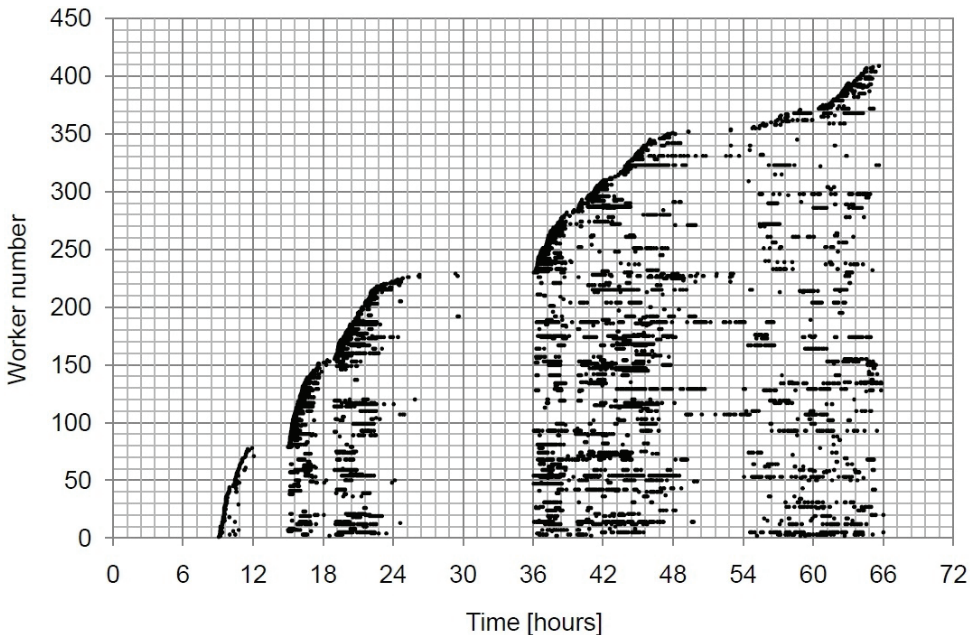


Figure 8.10.: Incoming worker responses during an experiment on the clickworker.com platform.

Full inspection

For the full inspection simulations, a quality goal φ_{min}, an escalation limit ε_{max} and the initial number of categories v need to be specified. The parameters have been set as follows: The initial number of categories was set to 2 because an average of 1.85 unique responses were provided per task in the gold standard data set. This indicates that there are typically about two different responses considered by the workers.

Quality goals of 0.95 and 0.99 have been tested. The escalation limit was initially set to 0.01 as recommended in section 5.5, for a second experiment it was increased to 0.09.

Table 8.6 summarizes the results. Each of the experiments represents an average of three simulations.

Table 8.6.: Performance of the CSP/DVM in the address research scenario (simulations).

Experiment		1	2	3	4
Parameters	Type	sim.	sim.	sim.	sim.
	Inspection	full	full	full	random
	$AOQL$	-	-	-	0.050
	Clearance number i	-	-	-	2
	Sample fraction f	1.000	1.000	1.000	0.612
	Quality goal φ_{min}	0.950	0.950	0.990	0.990
	Escalation limit ε_{max}	0.010	0.090	0.010	0.010
	Response categories v	2	2	2	2
	No gold standard	9.7%	9.7%	9.7%	9.7%
Results	Accuracy	96.7%	96.8%	98.2%	96.4%
	AFI	100.0%	100.0%	100.0%	80.6%
	Average redundancy	2.826	2.510	3.755	3.284
	Maximum redundancy	6	5	7	7
	Escalated	5.7%	14.9%	11.3%	9.6%
	No gold standard escalated	19.2%	41.2%	38.5%	34.0%

First, the results of the two full inspection experiments with a quality goal of 0.950 are discussed (experiment 1 and 2), which only differ in the escalation limit ε_{max}. With an accuracy of 96.7% and 96.8%, both experiments clearly exceed the quality goal. Experiment 1 ($\varepsilon_{max} = 0.010$) reaches an average redundancy of 2.826 and a maximum redundancy of 6 assignments per task. A fraction of 5.7% tasks was escalated. By increasing the escalation limit to 0.09, the redundancy decreases to 2.510 in average and to 5 at maximum because the escalation mechanism is reacting more sensitively. As a consequence, almost three times as many tasks are escalated (14.9%).

As discussed above, no gold standard was defined for 97 of the 1001 tasks because there had been no strong agreement among the workers contributing to the gold standard data set. The objective of the escalation mechanism is to identify such ambiguous tasks so the requester becomes aware of the issue and can take corrective actions. The last table row indicates which portion of the no-gold-standard tasks have been correctly identified by the escalation mechanism. This portion had been 19.2% for the escalation limit of 0.010 and it increased to 41.2% when switching to the more sensitive escalation limit of 0.090.

In the third full inspection experiment (experiment 3) which had a higher quality goal than the experiments 1 and 2, the DVM only just missed the quality goal by reaching a quality level of 0.982 instead of 0.990. This is actually not surprising when considering the way the gold standard was constructed, on which the assessment is based. Section 8.1.2 has revealed, that the DVM may lead even to a higher accuracy than a massive majority vote. Thus, for high quality goals, majority vote may not be an adequate benchmark for the DVM because the DVM outperforms the traditional majority vote.

Naturally, the higher quality goal resulted in a higher redundancy of 3.755 on average and 7 at maximum. A portion of 11.3% tasks was escalated, comprising 34.0% of no-gold-standard tasks.

Random inspection with CSP-1

For the random inspection experiment (experiment 4 in table 8.6), the DVM was complemented by the CSP-1. An $AOQL$ or 0.050 was chosen which corresponds to the quality goal of 0.95 in the full inspection experiments. The appropriate clearance number i and sample fraction f have been calculated according to section 5.5.1 by assuming an average incoming quality level p of 0.2, an inspection error e_1 and e_2 of 0.1 each, and an average run length of 16 tasks. According to the recommendations in section 5.5, the inspection quality φ_{min} was set to 0.99 and the escalation limit ε_{max} to 0.010.

First of all, with 96.4%, the quality goal of 0.950 was again clearly met. By adding acceptance sampling, the efficiency was supposed to increase in contrast to using the DVM only. However, compared to the corresponding full inspection experiment (experiment 1), the CSP-1 instead decreased the efficiency by increasing the average redundancy from 2.826 to 3.283. The increase is obviously caused by the very high AFI of 80.6%, because to all the inspected tasks, the inspection quality goal φ_{min} of 0.99 applies. Thus, more than 80% of the full inspection effort in experiment 3 is needed, in which the same parameters for φ_{min} and ε_{max} are used. Considering this inspection effort, the use of acceptance sampling is counterproductive for such a high AFI, i.e. in situations in which a large fraction of the tasks needs inspection.

But why is the AFI so much higher than it had been in the OCR scenario in section 8.1, where it only reached 41.0%? It is the number of contributing workers, that makes the difference here. There had been only 38 workers contributing to the OCR scenario while there had been 191[8] workers contributing to experiment 4 in table 8.6. Therefore, the average run length of the CSP-1 is only 16 tasks per worker. According to the clearance number i, the first 4 tasks are fully inspected for all workers. Then, the CSP-1 switches to random inspection and in the best case, if all workers provide correct responses, 61.2% of the remaining tasks will be inspected. This leads to a minimum AFI of $4 \times 100\% + 12 \times 61.2\%)/16 = 70.9\%$. Each time an incorrect response is identified by the inspection mechanism, the CSP-1 status of the corresponding worker will switch back to full inspection mode which further increases the redundancy. Thus, for a short experiment with a large number of workers, applying the CSP-1 does not make sense. Yet, the problem is to some extent an artifact of the simulation, because much more workers have contributed to the data set on which the simulation is performed than would contribute to a live experiment. Therefore, the random inspection mode will also be investigated in live mode.

8.2.4. Live experiment

The live experiments have been performed directly on the Clickworker platform using the QM plugin specifically developed for this case. After a number of pre-tests, a random inspection experiment was performed which is described here. Because of temporal limitations, no full inspection experiment was performed.

For the sake of comparability, the live experiment was conducted with the same parameters as the simulation experiment 4: The clearance number i was set to 2, the sample fraction f to 0.612, the inspection quality goal φ_{min} to 0.990 and the escalation limit ε_{max} to 0.010. Like for all address research experiments, the initial number of categories v was set to 2. A general difference of the live experiment compared to the simulations is that it is based

[8] This number reflects that not all available assignments have been needed for the simulations. Thus, not all the 409 workers who contributed to the simulation data sets also contributed to the simulations.

on such tasks only for which a gold standard had been defined. This decision was driven by the aim to reduce the execution effort and costs. The gold standard is a prerequisite for assessing the results returned by the QM mechanism. Therefore, no advantage was seen in generating results that cannot be evaluated. Later on, that decision was revoked for the simulations in order to determine the benefit of the escalation mechanism. Yet, the live experiment could not be simply be repeated because it had been implemented in the Clickworker environment. As a consequence, fewer tasks have been performed in the live experiment and as opposed to the simulation experiments, the analysis does not provide any feedback about the effectiveness of the escalation mechanisms.

Execution performance

As illustrated in figure 8.11, the experiment shows the typical execution progress that was already observed in the OCR live experiment in section 8.1.3. The number of workers increases almost linearly until 83 workers are contributing. After most of the assignments have been completed, the progress suddenly declines and the remaining responses slowly trickle in. In the last 1:20 hours, only 40 responses are arriving. For a productive use of the CSP/DVM, the slow down should be avoided according to the recommendations in section 8.1.3.

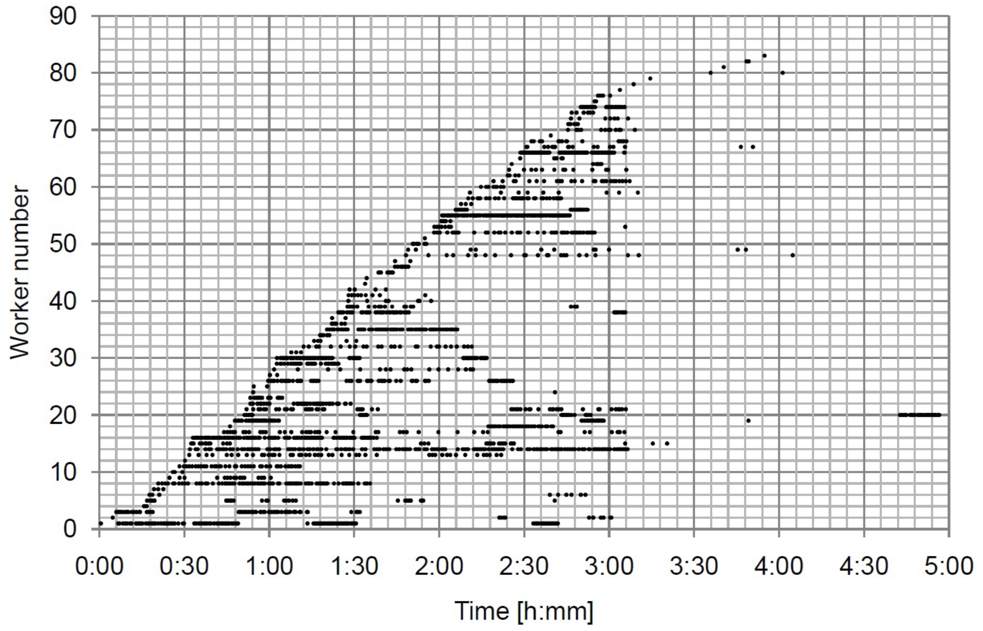


Figure 8.11.: Incoming worker responses during a live experiment with the CSP/DVM on the clickworker.com platform.

Random inspection with CSP-1

Table 8.7 provides the results of the live experiment. Most importantly, with an accuracy level of 96.2%, the target accuracy of 95% ($AOQL = 0.05$) is again clearly met. As expected, the AFI is much lower than for the corresponding simulation experiment 4 (64.8% compared to 80.6%). Likewise, the average redundancy per task was reduced from 3.284 to 2.166, which also considerably increases the efficiency compared to the full inspection experiments. Obviously, as opposed to the simulation, the live experiment did benefit from complementing the DVM with acceptance sampling.

Table 8.7.: Performance of the CSP/DVM in the address research scenario (live experiment).

Experiment		5
Parameters	Type	live
	Inspection	random
	$AOQL$	0.050
	Clearance number i	2
	Sample fraction f	0.612
	Quality goal φ_{min}	0.990
	Escalation limit ε_{max}	0.010
	Response categories v	2
Results	Accuracy	96.2%
	AFI	64.8%
	Average redundancy	2.166
	Maximum redundancy	7
	Escalated	6.1%

8.2.5. Discussion

The case study has shown that the CSP/DVM can be successfully applied to a business scenario and that it is capable of achieving predefined quality objectives in an efficient way. In doing so, the case study also revealed a series of important insights regarding the gold standard, the simulation mode of the CSP/DVM tool and the escalation process.

One specific challenge had been that no gold standard could be made available by Telegate. There had been no examples for good quality results and no experience regarding what level of quality can actually be achieved at all with a reasonable effort. Thus, it had been important to agree on a simple and intuitive process for generating a gold standard that could be utilized as a baseline for the experiments. The traditional majority vote turned out to be a good instrument for that. Telegate was convinced that if a large number of skilled workers agree on a specific response, there is a great chance that the response must be reasonable. This effect is certainly much more comprehensible than the CSP-1/DVM with all its formulas and parameters. However, the traditional majority vote suffers from inefficiency and therefore does not represent a desirable QM tool. The experiments of this section and in section 8.1 have demonstrated that the CSP/DVM can provide a response quality that is comparable to a massive majority vote or even outperforms it at a much lower execution effort. While each task was performed by 8 workers when generating the gold standard, an average of only 2.166 workers was needed per task by the CSP/DVM, which corresponds to a reduction of 73%. This insight provides a valuable argument for using the CSP/DVM.

The simulation mode of the CSP/DVM tool provided an effective way of investigating the effect of different CSP/DVM parameters without spending too much time and money on live experiments. When implementing the CSP/DVM in the Clickworker environment, the tool also served as a guideline and reference for validating the implementation. However, as the case study has shown, the simulation of random inspection experiments may lead to a higher AFI than live experiments. This happens if a large number of workers have contributed to the simulation data set and the average run length of the CSP-1 per worker becomes low. Therefore, it rather represents a deficiency of the simulation mode than of the CSP/DVM itself.

The case study has also demonstrated the effectiveness of the escalation mechanism. Depending on the parameters, a large portion of the ambiguous tasks are getting escalated and serve as a starting point for process improvements by the requester.

Altogether, the address research case study and the OCR model scenario have shown that the CSP/DVM can provide a reasonable QM tool for cloud labor services in both academic and business scenarios. In the following chapter, two additional case studies will be discussed.

9. Evaluation of model variations

The model variations developed in chapter 6 have been evaluated in two separate case studies. While the DVM for multi-labeling was applied to a *medical coding* business scenario, the model extension for non-deterministic tasks was applied to a *product research scenario*. In order to investigate the managerial implications of the models from the different perspectives of the basic cloud labor scenario, industry partners from all three sides have been included: A requester, a cloud labor platform and a workforce provider. Section 9.1 covers the *medical coding* case study while section 9.2 addresses the product research scenario.

9.1. Case study: Medical coding

Medical treatments can cause substantial costs for patients. Therefore, health insurance providers play an important role in healthcare systems as they cover at least a portion of the costs. The actual scope of benefits will depend on the type of insurance provider and on the contract. In Germany, there are two general types of health insurance providers, *statutory* and *private* ones. Private health insurance providers usually provide more benefits, but they are limited to customers whose monthly wage exceeds a certain minimum. While statutory health insurances are directly charged by the physicians, private health insurances are charged by the patients who receive the invoice from their physicians. Besides the name and address of the patient and the physician, the invoice comprises one or more medical diagnoses, the treatment that has been performed by the physician and the amount invoiced. When charging statutory health insurances though, diagnoses must be standardized according to the *International Classification of Diseases* (ICD) provided by the World Health Organization(WHO, 2013). When charging the patients of private health insurances, free-text diagnoses can be provided instead because the insurance cannot demand standardized codes to be provided. The lack of standardized diagnoses makes it difficult for the insurance company to validate the bills and to compile statistics. Thus, private health insurances aim to perform the coding step themselves which is expensive and error-prone because a huge number of invoices need to be processed and the coding step cannot be fully automated. Moreover, the number of invoices is fluctuating so a variable workforce is required for the manual coding effort.

Together with IBM and a private health insurance company, the medical coding problem has been identified as a promising use case for cloud labor services because it meets all the criteria listed in section 2.4.2: First, the medical coding task can be mapped to an electronic interface, by displaying the diagnoses on a screen. Second, there is no personal interaction needed with the requester because all required information can be provided along with the diagnoses. Thirdly, it is subject to a large and varying workload and finally, it cannot be fully automated.

This case study maps the medical coding problem to a cloud labor service scenario and applies the DVM for multi-labeling to manage the quality of the resulting ICD-10 codes. After illustrating the experimental setup in section 9.1.1, section 9.1.2 explains how the DVM for multi-labeling was applied. Section 9.1.3 then describes the actual experiments and sections 9.1.4 and 9.1.5 present and discuss the results. Finally, section 9.1.6 introduces an example application of the medical coding solution.

9.1.1. Experimental setup

Figure 9.1 illustrates how the problem was mapped to a cloud labor service task. The health insurance provider feeds the diagnoses taken from the physicians' invoices into a cloud labor service platform. The platform then turns them into tasks, which are made available to the workforce of a call center. The call center agents grab the tasks and turn the free text diagnosis into standardized ICD-10 codes with the help of an interactive tool supporting the coding step. The platform ensures the quality of the work results by applying the DVM for multi-labeling and finally returns the ICD-10 codes back to the health insurance company.

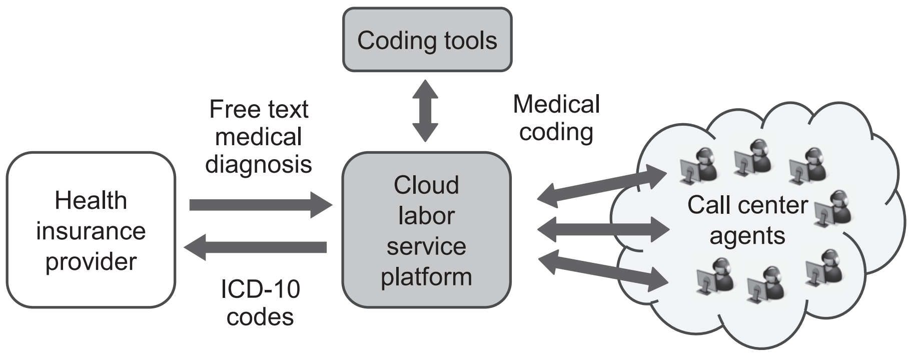

Coding tools
Free text medical diagnosis
Medical coding
Health insurance provider
Cloud labor service platform
Call center agents
ICD-10 codes

Figure 9.1.: Cloud labor service scenario for coding of free text medical diagnoses.

The case study was performed in cooperation with a number of industry partners:

- A medium-sized private health insurance provider (HIP[1]) located in Germany. As insurance services are rather a commodity business, providers are naturally looking for opportunities to differentiate from their competition. Therefore, the partner has seen the study mainly as an opportunity to develop a starting point for new innovative services. Another objective was to reduce costs.
- buw Group[2] (BUW), the largest owner-managed call center business in Germany, provides customer care solutions across multiple channels ranging from traditional

[1] As the name of the company cannot be disclosed, the abbreviation HIP for *health insurance provider* is used here.
[2] `http://www.buw.de/en/startseite.html`

call center operations to complex business process outsourcing, complemented by consultancy services. More than 4,400 employees are working at eight locations in Germany, Hungary and Romania. BUW is proud of repeatedly having been awarded the prize for the most favored employer in its area (Buw Group, 2013). BUW's primary intention in evaluating the cloud labor concept has been the hope for an opportunity of achieving a more uniform utilization of their workforce and of advancing their traditional business model.

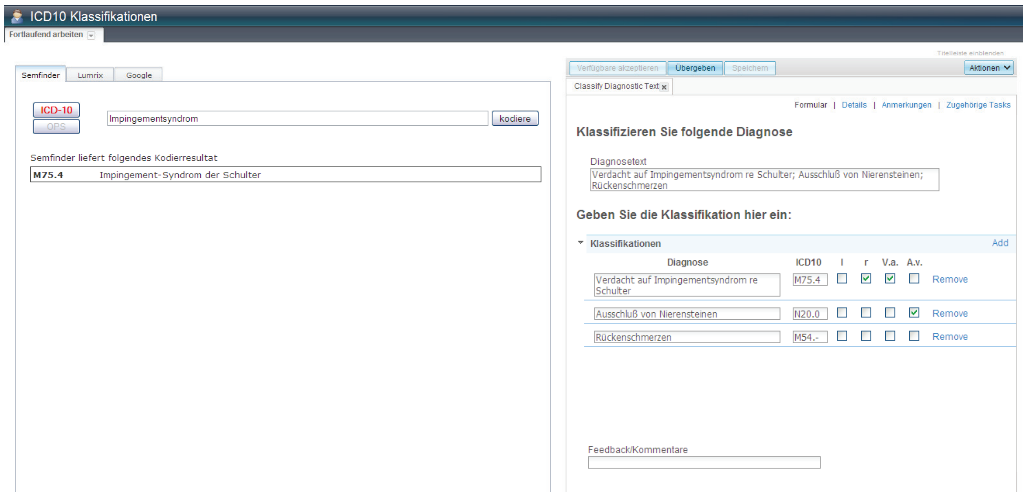
- Semfinder AG[3], a small-scale Swiss company located in Kreuzlingen, develops solutions for the semantic interpretation of free text information with a specific focus on the classification of medical diagnoses according to the *International Classification of Diseases* (ICD). Semfinder's services are used by more than 400 hospitals in Germany, Austria and Switzerland (Semfinder AG, 2013). The company has seen the study as a potential opportunity for extending their scope to new applications of their coding software.
- The IBM Global Business Services[4] insurance team in Germany combines industry specific skills with IT oriented expertise by offering consulting and implementation services, insurance specific components and architectures, as well as complete solutions (IBM, 2008). The IBM team is continuously seeking opportunities to deliver new, innovative solutions to their customers.
- IBM Research and Development GmbH in Germany is one of the largest IBM research and development centers located outside the US. About 2,000 engineers and natural scientists are working on more than 70 projects, ranging from hardware, firmware and operating system development to cloud computing, analytics, BPM, data center optimization and web technologies (IBM, 2012). The study was supported by the BPM team, who sees cloud labor services as an opportunity for extending the reach of their technology.

For performing the actual coding task, a Web interface was offered to the call center agents, displaying the free text diagnoses and a feedback form along with the specialized interactive web-based tool Semfinder[5]. The Semfinder tool was integrated into the Web interface and provided the medical expertise for an accurate coding of the medical diagnoses while the service agents had to operate the tool properly and to feed it with the right information. Especially in case of misspelled or colloquial diagnoses the agents had to perform additional research in order to transform the diagnosis text into a form that the tool was capable of understanding. For this purpose, additional research tools like Lumrix[6] and Google were made available within the Web application. No specific medical expertise was required of the service agents.

In order to speed up operation, drag and drop functionality could be used to move the output of the Semfinder tool into the response form of the Web application. Figure 9.2 shows a screenshot of the Web application used by the call center agents. On the left, the three research tools (Semfinder, Lumrix and Google) are available as tabs, while on the right, the response form is provided, into which the worker enters the ICD-10 codes, along with the portions of the free text that represent an individual code. Additional flags can be set that indicate that a symptom applies to the right or left side of the patient, whether a

[3] `http://www.semfinder.com/en/home.html`
[4] `http://www-935.ibm.com/services/de/gbs/consulting/` (German), last accessed on April 2, 2013.
[5] `www.semfinder.com/en`
[6] `http://www.lumrix.de/`

diagnosis is just suspected (German: ”Verdacht auf” (V.a.)) or whether a specific diagnosis can be excluded (German: ”Ausschluss von” (A.v.)). Although feedback about the flags had been requested from the workers, the actual experiment did not regard the flags, but focused on the actual ICD-10 codes only.



Impingementsyndrom

Semfinder liefert folgendes Kodierresultat

M75.4 Impingement-Syndrom der Schulter

Klassifizieren Sie folgende Diagnose

Diagnosetext

Verdacht auf Impingementsyndrom re Schulter; Ausschluß von Nierensteinen; Rückenschmerzen

Geben Sie die Klassifikation hier ein:

Klassifikationen

Diagnose	ICD10	l	r	V.a.	A.v.
Verdacht auf Impingementsyndrom re Schulter	M75.4	☐	✓	✓	☐
Ausschluß von Nierensteinen	N20.0	☐	☐	☐	✓
Rückenschmerzen	M54.-	☐	☐	☐	☐

Figure 9.2.: Screenshot of the product research scenario data acquisition task interface (German).

9.1.2. ICD-10 coding as a multi-labeling task

ICD stands for the *International Classification of Diseases*, a classification of the World Health Organization (WHO, 2013) which represents ”the standard diagnostic tool for epidemiology, health management and clinical purposes” that is used to classify diseases and other health problems. The version that was used for the use case is the 10th revision of the ICD, the so-called ICD-10 (WHO, 2010). An example for a diagnosis text is ”fracture of the metatarsal bone”, which corresponds to the ICD-10 code ”S92.3”.

ICD-10 coding is a multi-labeling task because one or more codes (labels) are assigned to each diagnosis text. However, as opposed to the assumptions of the DVM for multi-labeling made in section 6.1.2, the ICD-10 represents a hierarchical rather than a flat structure and only has a limited binary relevance. Therefore, some codes are not required, but they also do not hurt.

Because of the hierarchical structure, the coding task is not a fully deterministic task, as there may be multiple sets of codes that represent a correct result. Also, there may be dependencies between labels. The Semfinder coding tool helped to overcome these challenges by returning reproducible results, even if there are multiple correct sets of codes. This also satisfied the needs of the voting mechanism. The limited binary relevance was compensated by considering only required codes.

An additional challenge of the ICD-10 coding scenario is that there is a huge number of about 14,000 codes, of which only about 10 are typically associated to a diagnosis text. Therefore, a bad worker who returns a large number of incorrect responses would still achieve a specificity of close to 1. In order to avoid this issue, only the labels actually returned by any of the workers were considered to exist, i.e. only those labels are regarded in the response matrix (6.1).

9.1.3. Task execution

The experiments have been performed on a sample of 104 free text diagnoses extracted from physicians' invoices that customers had submitted to the private health insurance company for reimbursement in August 2010 (59 diagnoses) and February 2011 (45 diagnoses). Using the prototypical cloud labor service platform, seven redundant labeling decisions were collected for each of the diagnosis texts from a group of 10 participating call center agents. The gold standard was identified by an ICD-10 coding expert of the insurance company by manually validating all responses of the call center agents. The labeling decisions for the first 59 diagnosis texts were used for determining the initial sensitivity and specificity of the agents. The labeling decisions for the remaining 45 diagnosis texts were used for validating the DVM for multi-labeling. For this purpose, the toolkit introduced in chapter 7 was extended. The tool was used in simulation mode, i.e. the worker's feedback was retrieved from the pool of available labeling decisions whenever the mechanism decided that redundancy should be increased.

9.1.4. Results

Based on the first 59 diagnosis texts and the gold standards defined by the coding expert of the health insurance company, an average initial sensitivity of 0.74 and a specificity of 0.89 were determined for the participating call center agents, i.e. in average, the agents correctly identified 74% of the required ICD-10 codes and they omitted 89% of the codes that should *not* be returned. The good performance of the agents confirms that the setup is capable of generating considerably good coding results, even though the call center agents didn't have any specific medical background.[7]

In order to check whether the DVM for multi-labeling is capable of achieving a well-defined correctness goal, a sensitivity and specificity goal of 0.90 was assumed. The minimum level of redundancy was set to 2. As the labeling performance of the individual workers is updated after every successful task, the order of tasks and workers may affect the quality of the obtained final results. Therefore, five experiments with a random order of tasks and workers have been performed. The averaged results are presented in the last row of table 9.1. An average of 42.4 out of the 45 tasks were completed while 2.6 tasks were not

Table 9.1.: Performance of the DVM for multi-labeling with a sensitivity and specificity goal of 0.90 each applied to a medical coding scenario.

Experiment number	Average sensitivity	Average specificity	Number of results	Average redundancy	Unfinished tasks
1	0.904	0.906	43	2.400	2
2	0.912	0.941	41	2.467	4
3	0.902	0.904	43	2.489	2
4	0.906	0.904	43	2.467	2
5	0.912	0.937	42	2.467	3
Average	0.907	0.918	42.4	2.458	2.6

[7] Preliminary experiments have been conducted that indicate that a comparable worker performance can be achieved on the MTurk platform. However, it had been challenging to find enough German speaking participants who are willing to work on the diagnosis scenario. Even after increasing the payment drastically, only about a dozen workers contributed to the experiment.

completed due to an insufficient number of responses available for the simulation. These results were not included in the assessment of the quality but in the assessment of the average redundancy.

The DVM for multi-labeling resulted in an average of 2.458 workers being involved in each of the labeling tasks. With an average sensitivity of 0.907 and an average specificity of 0.918, the sensitivity and specificity objective of 0.90 could be achieved.

9.1.5. Discussion

The results indicate that the DVM for multi-labeling is capable of meeting certain predefined sensitivity and specificity objectives. Even more importantly, the scenario shows that cloud labor services can be a valid instrument for enabling a new business model. The cooperation of the health insurance provider, the call center and the technology providers generates an ecosystem in which value is generated for each of the contributors:

The *health insurance provider* benefits from correct and complete coding of medical diagnoses in multiple ways. Most importantly, understanding the patient's diseases allows for developing a holistic view on them. Specific treatments, physicians and hospitals can be recommended and their success can be assessed. That way, recovery can be accelerated and costs can be reduced. Furthermore, tailored services can be designed for individual customers. For example, flexible benefits can be provided even to customers who belong to a high-risk group. This can be achieved by recognizing consequences of previous diseases based on the ICD-10 codes and by explicitly excluding them from the contract. Another important use case is the detection of fraud, e.g. the prescription of unnecessary treatments. This problem is enforced by the coexistence of private and statutory health insurances in Germany. While private health insurances are rather generous regarding the expenses they reimburse, the statutory health insurances are continuously reducing their coverage. If in a family, one partner's statutory health insurance doesn't cover a treatment any more, physicians sometimes put it on the invoice of the other partner's private health insurance. Thanks to the ICD-10 codes, situations like that can be identified because the treatments do not match the person's actual symptoms, i.e. by comparing the treatments prescribed by the physicians with the patient's actual diagnoses, unnecessary treatments can be identified.

For the *call center* the primary advantage is an increased utilization of the workforce. Especially at noon time there is a period of two hours from 12 pm to 2 pm during which the call center agents are highly under-utilized because there are only few incoming calls and customers don't want to be called. Complementing the synchronous phone business with asynchronous cloud labor services can close this gap. Because the call center agents are paid anyway, the service can be provided at competitive rates. Working with call center agents can even be less expensive than working with Internet users. The approach can also increase the employee satisfaction. The call center agents who are used to speaking on the phone all day appreciated the job variation generated by the cloud labor services. The maybe most important, strategic benefit of the concept is that it enhances the traditional business model of the call center given that the revenue made with the traditional phone business can be expected to decrease over time.

For the *technology providers* IBM and Semfinder, the ICD-10 coding solution represents an opportunity to establish new technology and services. In fact, by partnering with Semfinder, IBM has turned the solution into a commercial service which is used by several private health insurance providers in order to process millions of medical diagnoses per year. The service extends the existing *Insurance Service Hub* offered by IBM which is introduced in the following section.

9.1.6. IBM Insurance Service Hub

Building on the success of the case study, IBM has commercialized the medical coding solution by adding it to their so called *Insurance Service Hub* (ISH) offering. This section introduces the ISH and describes how it has been complemented by the medical coding service.

As mentioned in the introduction of this chapter, private health insurances are not charged directly by the physicians, but by the patients who receive the invoices from the physicians and submit it to the insurance company for refund. This process results in a high effort for the insurance company, given that the invoices are still on paper and need to be scanned and processed with OCR software. The process is also prone to errors and fraud because the amount numbers can be manually changed on the paper.

The ISH exploits the fact that most of the invoices are actually not sent by the physicians but by a relatively small number of so called *national associations of statutory health insurance physicians* (NASHIP) who are providing billing services to the physicians (Gräfen, 2012). On behalf of IBM, the associations print dot-matrix codes on the invoices that represent the patient information and provide a reference to the actual invoice data. When receiving the invoices from the patients, the insurance company does no longer have to process the document by OCR but just needs to scan the dot-matrix codes which is much less error-prone. Based on the invoice ID, the actual invoice data can then be retrieved from the ISH, which in turn receives it from the corresponding NASHIP. The risk of fraud is reduced because manual changes on the invoice do not affect the process. Figure 9.3 illustrates the basic schema of the ISH.

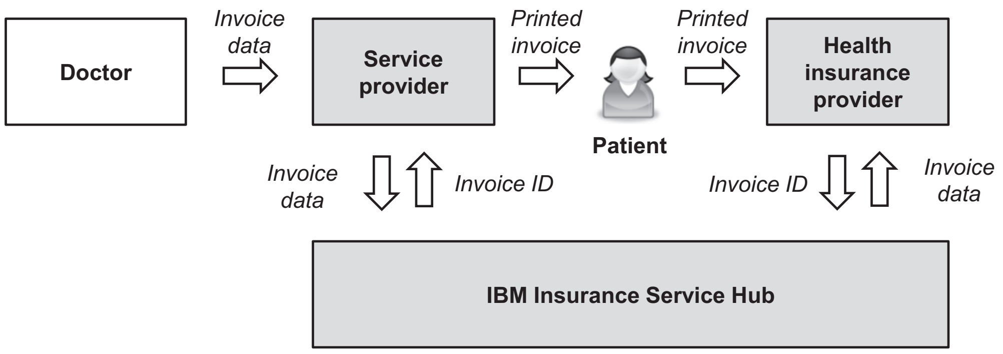


Figure 9.3.: Basic schema of the IBM Insurance Service Hub, adopted from Gräfen (2012).

An optional "people cloud" feature of the ISH allows for implicit coding of medical diagnoses, i.e. free text diagnoses on the invoices are turned into ICD-10 codes by the ISH using a combination of automatic coding with the Semfinder tool and manual coding with cloud labor services. The solution ties in with the scenario and the QM approach developed in the case study. The successful application of cloud labor services in the case study and in the IBM ISH solution confirms the viability of the approach.

9.2. Case study: Product research

In the *product research* case study, the model variation for non-deterministic tasks is applied, which was introduced in section 6.2. It is represented by the *group validation approach*. As opposed to the DVM it is not based on the voting but on the validation pattern. From an SQC perspective, the approach does not use continuous sampling plans but acceptance sampling.

The case study has been evaluated in a business scenario together with the German cloud labor platform provider Bitworxx[8]. Bitworxx corporation was founded at the end of 2008 in Düsseldorf, Germany. Rather than on private Internet users, the platform relies on the workforce of a worldwide network of partners, mainly call centers and business process outsourcers, who naturally struggle with overcapacities in times of low utilization. This has multiple advantages: On the one hand it lowers the costs, on the other hand, because such partners are prepared to assure high privacy and security standards, it enables Bitworxx to deal with confidential customer data. (Bitworxx GmbH, 2013)

In the case study scenario, the service requester was represented by an online shopping platform on which traders can sell products with minimal overhead. Rather than taking pictures of their products and creating product descriptions with high effort, the traders only need to specify the product's so called *Global Trade Item Number* (GTIN).

GTIN is a worldwide standard for uniquely identifying trade items like products and services, which is released by the GS1 non-profit association that has member organizations in more than 100 countries. GTIN is represented by an up to 14 digit number, which is used as a bar code on product packages or tags, and which refers to item related information stored in databases. GTIN comprises former regional standards like the *European Article Number* (EAN)[9] or the US American *Universal Product Codes* (UPC). (GS1, 2006)

Even though GTIN uniquely identifies a product, there is no central directory that keeps the product information. GS1 only maintains a catalog of the company prefixes while the companies themselves are responsible for allocating the remaining digits of the GTIN. One way to identify the product information is to perform a GTIN research on the Internet. The shopping platform achieved this aim by outsourcing the task to the Bitworxx platform. The expectation was to receive the task results at the latest after one working day.

9.2.1. Experimental setup

The initial task interface presented to the user contains the task description, the EAN to be researched as well as a number of radio buttons for selecting a product category. Once a category is selected, corresponding input fields are displayed for entering the actual product details. For example, if the category *music* is selected, the name of the album and the interpreter may be entered in the next step. If no information is found for the EAN, this may be indicated by a checkbox. Figure A.2 in appendix A provides a screenshot of the original German data acquisition task interface.

The EAN research scenario represents a *non-deterministic* task according to the definition in section 3.1.2, because there may be multiple correct ways to specify the product category and the product name. For example, the exact same product is called "InLine®USB 2.0 Hub, 4 Port, LED Hub, black" by one dealer and "InLine 33294L USB 2.0 Hub (4-Port,

[8] `http://www.bitworxx.com/`

[9] The term EAN is still commonly used and has therefore been used for the case study as well.

LED Hub) black" by another. Both names can be considered correct because even the manufacturer's website[10] does not provide an *official* name. Therefore, the core model introduced in chapter 5 cannot easily be applied here. For this reason, the validation pattern is used in which the responses returned by a worker are validated by other workers in a separate validation task.

The user interface of the validation task is identical to the filled-in research screen, except that there are additional fields for entering the actual review. For indicating one of three error types one of three check boxes can be selected. The scenario differentiates between the following three types of errors that a worker may perform: Error *"no information found"* means that the worker did not find the requested EAN even though it could be found by the reviewer. Error *"non-conforming product category selected"* means that the worker did not specify the correct category for the product, and *"incorrect product details specified"* means that not the correct product details have been provided. Additional input fields allow for passing back feedback to the original worker. As mentioned in section 6.2.3, it is assumed that the worker can correct the problem based on the feedback provided by the reviewers. Figure A.3 in appendix A shows a screenshot of the original German review task interface. Combining the error rates of all three error types, the maximum error rate was expected to be in the area of 10-15%.

Bitworxx already had a quality management procedure in place before the case study was started. It simply consisted of a 100% inspection performed by a Bitworxx employee. In case of incorrect responses, individual feedback was passed to the workers to help them getting better over time. The review is also regarded as a way to communicate the requirements to the workers. Despite the high review effort and costs, Bitworxx had made positive experience with this procedure, as the customers' quality requirements always had been met. Because of that, it was used as the baseline and reference for the case study. The result of the Bitworxx review was used as the gold standard for assessing the QM mechanism.

9.2.2. Task execution

The data for the experiment was collected within the real EAN research project from January 20 to 28, 2010. Initially, a batch of 2002 research tasks was processed as usually and was reviewed by a Bitworxx employee. For each research task, six additional validation tasks were processed by the crowd on seven working days, which resulted in a total of 12,012 validation tasks. The workforce consisted of ten German call center agents who had been experienced with the EAN scenario. Nine of them acted as workers and eight as reviewers. They have not been aware that they supported a scientific experiment, nor that the same validation tasks were performed by multiple reviewers. Worker error rates were around 10-15%.

Table 9.2 shows the number of research tasks that have been performed per worker and per day. The *worker ID* refers to the internal id used at Bitworxx. The tasks per day are summed up in the last column while the tasks per worker are summed up in the last row. While more than one third of the research tasks have been performed by a single worker (worker 102), there is a more balanced picture for the validation tasks as most of the participating reviewers have completed a similar number of them. Table 9.3 summarizes the validation tasks per day and worker. As the table reveals, three of the workers have

[10] `http://www.inline-info.de/index.php?lang=en`, last accessed on November 1, 2013.

Table 9.2.: Number of research tasks submitted in the product research scenario per day and per worker.

Worker ID	101	102	103	104	105	106	107	180	206	Total
Day 1	59	-	-	15	-	-	25	7	-	106
Day 2	13	151	4	-	-	22	24	18	-	232
Day 3	-	165	24	151	29	-	1	85	-	455
Day 4	206	41	16	35	85	30	120	48	13	594
Day 5	8	199	23	-	36	-	1	52	2	321
Day 6	2	235	-	-	-	-	-	1	-	238
Day 7	1	54	-	-	-	1	-	-	-	56
Total	289	845	67	201	150	53	171	211	15	2002

Table 9.3.: Number of validation tasks submitted in the product research scenario per day and per worker.

Worker ID	101	103	104	105	106	107	180	191	Total
Day 1	106	106	106	106	-	106	106	-	636
Day 2	232	89	232	232	184	232	191	-	1392
Day 3	455	251	455	444	323	455	347	-	2730
Day 4	594	377	594	477	528	594	395	5	3564
Day 5	321	169	321	314	319	321	161	-	1926
Day 6	238	191	238	209	238	238	76	-	1428
Day 7	56	54	56	29	56	56	29	-	336
Total	2002	1237	2002	1811	1648	2002	1305	5	12012

reviewed all 2,002 tasks, which means that they have also reviewed the ones that have been completed by themselves. Because of technical limitations, this could not be easily avoided in the Bitworxx environment. However, it can be argued that even reviewing one's own contributions may lead to an improvement. Sometimes, it is helpful to follow-up on a piece of work with a fresh mind. And indeed, in some cases reviewers have rejected their own responses. Apart from that, they had not been aware of the fact that they would review their own contributions.

The actual quality management experiments have been performed as a simulation on the results of the validation tasks, including the ones performed by the Bitworxx employee. As long as less than six validations were needed for a response, the ones performed by the original worker were not included.

The simulations have been performed with a Java tool that had been developed specifically for that purpose. The individual components of the model are represented by separate Java classes. A file-based data store provides access to the raw data from the experiments at Bitworxx. The run-time parameters of the tool are provided as constants within the source code. Output reports are automatically generated as comma separated files that can be viewed and processed with spreadsheet tools like Microsoft Excel.

9.2.3. Full inspection

Initially, the group validation approach has been evaluated in full inspection mode based on the data described in the previous section. The objective of the experiment is to test the effect of the inspection mechanism.

Full inspection mode means that 100% of the responses are validated using the EM algorithm. For the group validation model, the primary output of the EM algorithm is an estimate T_z for the true response quality of each task z that decides whether a response is considered to be correct or incorrect. If an incorrect response is found, the task is assumed to be successfully reworked. Therefore, in the outgoing batch of responses, only those are incorrect that have not been detected by the inspection process, i.e. if the inspection process has made a type II error. Because the results had to be delivered within one day, a separate batch was created for each day. That way, the estimations generated by the EM algorithm on one day could be passed to it as an initial estimate for the next day.

Table 9.4 summarizes the performance of the full inspection mode for the seven days of the experiment. The last column represents the totals for the entire time period. The rows *actual correct* and *actual incorrect* refer to the assessment of the Bitworxx employee, which was used as the gold standard. *Found by inspection* represent the number of tasks correctly identified to be correct or incorrect by the inspection process. The type I and type II errors are calculated based on the deviations between the actual and the estimated values. The number or reworked responses comprises those that have been successfully identified to be incorrect plus the correct ones that have been found to be incorrect due to a type I error, e.g. on day 1 a number of $45 = (84 - 54) + 15$ responses had to be reworked. Finally, the *fraction defective* is the portion of actual incorrect tasks while the *outgoing quality* represents the portion of incorrect responses being delivered to the requester.

Table 9.4.: Performance of the group validation approach in full inspection mode.

	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Total
Total tasks	106	232	455	594	321	238	56	2002
Actual correct	84	176	408	559	296	237	38	1798
Found by inspection	54	164	376	514	276	229	36	1649
Type I error	35.7%	6.8%	7.8%	8.1%	6.8%	3.4%	5.3%	8.3%
Actual incorrect	22	56	47	35	25	1	18	204
Found by inspection	15	41	40	28	16	0	7	147
Type II error	31.8%	26.8%	14.9%	20.0%	36.0%	100.0%	61.1%	27.9%
Reworked	45	53	72	73	36	8	9	296
Percentage reworked	42.5%	22.8%	15.8%	12.3%	11.2%	3.4%	16.1%	14.8%
Fraction defective	20.8%	24.1%	10.3%	5.9%	7.8%	0.4%	32.1%	10.2%
Outgoing quality	6.6%	6.5%	1.5%	1.2%	2.8%	0.4%	19.6%	2.8%

Even though the type II error rate is high, the outgoing quality of 2.8% clearly falls below the requested quality level of 10-15% because the fraction defective is already at a low level.

One thing that catches one's eye is that the type I error is extraordinary high on day 1. This can be assumed to be caused by the fact that there had been no estimates available from previous runs of the EM algorithm which naturally affects its output. That is consistent with the other estimates made by the EM algorithm, the fraction defective and the type I and type II error rates of the reviewers.

Figure 9.4 compares the estimated fraction defective p with the actual values determined based on the gold standard which have already been provided by table 9.4. Interestingly, on day 1, the EM algorithm drastically overestimated the actual fraction defective. More than every second response considered incorrect had actually been correct. On day seven, p was underestimated, which may simply because there had been relatively few responses to be validated.

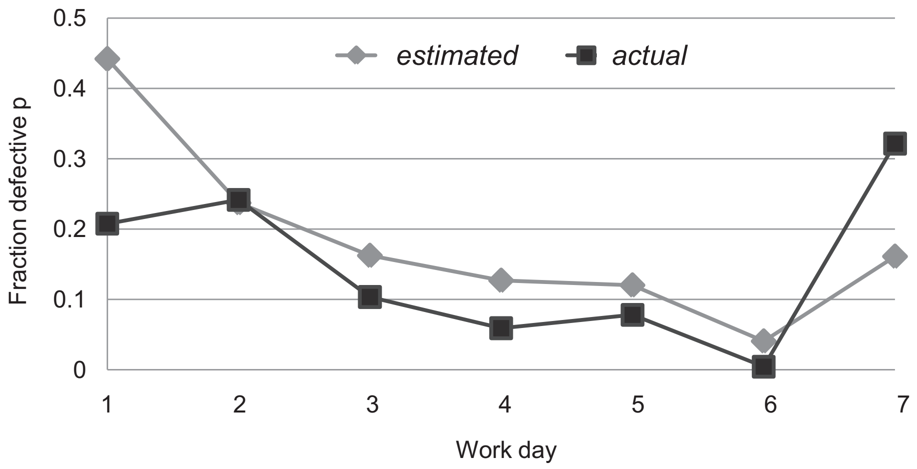


Figure 9.4.: Estimated vs. actual fraction defective in the full inspection mode of the group validation approach.

Figure 9.5 gives a similar picture for the weighted averages of the estimated type I and type II reviewer error rates e_1 and e_2 which are compared with the actual error rates determined based on the gold standard. On day 1, e_1 is highly overestimated, which is again consistent with the outlier of the overall type I error on day 1. However, it is important to understand that the individual error rates of the reviewers might be much higher than the overall error rates because the EM algorithm weights the judgments of bad workers lower. This becomes obvious when looking at the actual average type II error of the workers: With about 0.51 compared to 0.28 (see table 9.4) it is much higher than the average type II error of the EM algorithm.

The error rates of the reviewers are being investigated in more detail. Table 9.5 shows the distribution of the individual type I and type II according to the number of reviewers that have performed the same error. For example, there were seven situations in which five reviewers[11] voted for rejecting a response even though it was accepted by the Bitworxx reviewer. For some 21% of the responses, the majority of the reviewers decided differently than he did. This shows that there is a general gap between the perception of the Bitworxx reviewer compared to the other reviewers. There had even been situations in which the former one rejected a response, while all others accepted it. Possibly, the task requirements have not been communicated properly or have not been internalized by the reviewers to the same extent as by the Bitworxx employee, who knows the customer requirements at first hand. This is not surprising as for him, processing the validation responses is the daily business, while it had been a new experience for the others.

[11] That results in $7 \cdot 5 = 35$ individual votes.

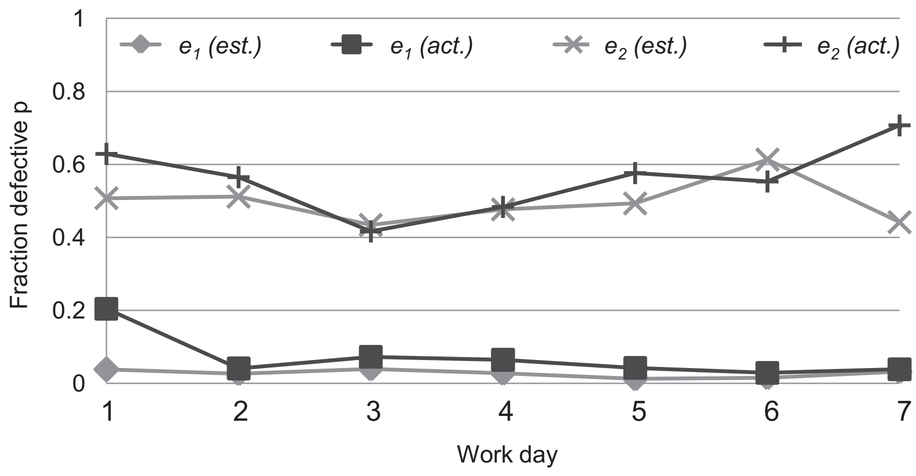


Figure 9.5.: Estimated vs. actual type I and type II error rates in the full inspection mode of the group validation approach.

Table 9.5.: Distributions of the actual type I and type II reviewer errors with regard to the agreement between the reviewers.

Votes	1	2	3	4	5	6	Total
Type I errors	19	35	46	32	29	26	656
Type II errors	227	65	55	27	7	0	665

However, the opposite case has been observed as well. Table 9.6 provides an example of the decisions and the feedback from six reviewers, five of them recommended to reject a response whereas the Bitworxx reviewer decided to accept it. From the feedback and by manually validating the task it becomes obvious that the latter one made a mistake. While the worker claimed that no information on EAN 21165105287 can be found, the reviewers identified a number of common websites that point to the respective product. Note that

Table 9.6.: Reviewer rating for task id 77176 (translated from German).

Reviewer	Rating	Comment
Bitworxx	correct	
101	incorrect	##1##http://www99.shopping.com/xPO-Saitek-Saitek-Mini-Color-UFO-Hub-Metallic-Blue##1##
107	incorrect	##1##http://www1.shopping.com/xPO-Saitek-Saitek-Mini-Color-UFO-Hub-Metallic-Blue##1##
104	incorrect	##1## Please also use Google for data research ##1##
105	correct	
180	incorrect	##1##http://www.shopping.com/xPO-Saitek-Saitek-Mini-Color-UFO-Hub-Metallic-Blue ##1##
106	incorrect	##1## Infos found with Google ##1##

the total number of votes is about the same for both error types even though the type II error happened much more often. This is because there are much more correct responses than there are incorrect ones. Therefore, the type II errors relate to a much lower number of responses, which results in a higher error rate.

9.2.4. Sample-based inspection

The sample-based inspection was performed according to section 6.2.4. The basic objective is to reduce the review effort by only inspecting a subset of the responses delivered by the workers. The experiments have been performed on the complete set of 2002 tasks that were formed into 38 lots, one per worker and per day, i.e. each lot consisted of the responses delivered by a single worker on a single day. According to table 9.2 this results in 38 lots.

For each lot, a separate sampling plan had been defined by looking up the sampling size and the acceptance number in table 4.1 for the appropriate lot size N. The $AOQL$ was set to 0.1 in accordance with the requester expectations. Based on Bitworxx' previous experience with the workers, the process average p_t was assumed to be 10%. While the sampling process must be performed per worker, the EM algorithm for the actual inspection process can be performed per day by mixing the samples taken from different workers. The number of reviewers was dynamically determined according to section 6.2.6 using the estimated type I and II error rates of the reviewers from the previous day. On day 1, the maximum available number of 6 reviews was used for all lots because no estimations were available. The reviewers were randomly picked from the available data in order to simulate the crowdsourcing scenario in which reviewers would pick autonomously from available tasks. To average out the random effects, the actual evaluation is being performed on an average of 5 experiments. A summary of the results is presented in table 9.7.

Table 9.7.: Overall performance of the group validation approach (average of 5 experiments).

Total tasks	2002
Lots	38
AOQL	0.1
Avg. fraction inspected	48.2%
Inspected by samples	335
Rejected lots	20
Inspected by full inspection	630
Reviews	5439
Avg reviews per inspection	5.64
Avg reviews per task	2.72
Considered incorrect / reworked	211
Percentage reworked	10.5%
Actual incorrect	204
Found by inspection	122
Not found by inspection	82
Incoming fraction defective	10.2%
Outgoing fraction defective	4.10%

Observations

Out of the 38 samples, 20 have been rejected. Because many of them only consist of a few tasks, the portion of corresponding tasks is much lower, but still represents 32.5% of the overall tasks. The average fraction inspected (AFI) was 48.2%, which means that almost every second response has been validated. Out of this, 335 inspections have been part of the regular sampling process while 630 were needed for the full inspection of the rejected lots. As each of the inspections was performed by an average of 5.6 reviewers, the overall number of reviews sums up to 5439, which corresponds to an average of 2.7 reviews per task. A number of 211 responses (10.5%) was considered to be incorrect by the inspection process so the responses had to be reworked. Out of them, 122 are indeed incorrect according to the gold standard. The remaining 82 incorrect responses have not been identified, which results in an outgoing fraction defective of 4.2%. A number of 89 responses are supposed be reworked even though they are correct according to the gold standard.

9.2.5. Discussion

The results confirm that the sample-based inspection can drastically reduce the review effort while still clearly meeting the quality objectives. Compared to the full inspection, the review effort was cut into less than half (48.2%) whereas the outgoing fraction defective increased from 2.8% to 4.1%. An *AFI* of 48.2% is certainly still large, the biggest part of it, however, is caused by the full-inspection of the rejected lots. Interestingly, the process turned out to be reproducible with regard to the decision whether to accept or reject a lot. The vast majority of the lots (19 out of a maximum of 21) was rejected by all 5 experiments. Table 9.8 illustrates the relation between the actual worker error rates determined by the gold standard and the rejected lots, for which the error rates are printed in bold. The averages per worker and per day are weighted according to the number of tasks that the worker has performed on the day (see table 9.2). For most lots,

Table 9.8.: Actual worker error rates per day according to the gold standard.

Worker ID	101	102	103	104	105	106	107	180	206	Avg.
Day 1	**0.20**	-	-	**0.20**	-	-	**0.28**	0.00	-	0.21
Day 2	**0.69**	**0.17**	**0.50**	-	-	**0.32**	**0.42**	0.11	-	0.24
Day 3	-	**0.22**	**0.04**	0.01	0.07	-	**1.00**	**0.06**	-	0.10
Day 4	0.01	**0.20**	0.06	0.03	0.09	**0.03**	0.09	0.04	0.00	0.06
Day 5	**1.00**	0.03	0.09	-	**0.11**	-	**1.00**	0.04	**1.00**	0.08
Day 6	0.00	0.00	-	-	-	-	-	1.00	-	0.00
Day 7	0.00	**0.31**	-	-	-	1.00	-	-	-	0.32
Avg.	0.11	0.11	0.09	0.03	0.09	0.17	0.18	0.06	0.13	0.10

there is an obvious correlation between error rate and rejection, but for some there is not. For instance, the contributions from worker 180 on day 6 were accepted even though the actual fraction defective had been 100%. Although, this is less surprising when realizing that the lot consisted of a single task only, but it may still appear strange that a completely corrupted lot is accepted. Another example is worker 106 who has performed 30 tasks on day 4 with an error rate of only 0.03. Anyway, the lot was rejected in all experiments. The explanation for both cases is simple: The crucial factor for the decision of the sampling process is not the gold standard but the feedback of the reviewers, which is consolidated by the EM algorithm. Table 9.9 illustrates that characteristic by attaching fictitious error rates to the lots that have been determined by the full inspection experiment described in the previous section. From the reviewer perspective, worker 180 did a good job on day 6, while worker 106 on day 4 did not. The averages are again weighted according to the number of tasks performed and numbers printed in bold represent rejected lots. Depending

Table 9.9.: Fictive worker error rates defined according to the reviewer decision in the full inspection mode of the group validation approach.

Worker ID	101	102	103	104	105	106	107	180	206	Avg.
Day 1	**0.44**	0.00	0.00	**0.27**	0.00	0.00	**0.64**	0.00	0.00	0.43
Day 2	**0.54**	**0.15**	**0.50**	0.00	0.00	**0.27**	**0.54**	0.17	0.00	0.23
Day 3	0.00	**0.27**	**0.17**	0.03	0.07	0.00	**1.00**	**0.20**	0.00	0.16
Day 4	0.07	**0.20**	0.13	0.00	0.24	**0.13**	0.18	0.13	0.08	0.13
Day 5	**0.63**	0.06	0.09	0.00	**0.36**	0.00	**1.00**	0.00	**1.00**	0.11
Day 6	0.00	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04
Day 7	0.00	**0.17**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.16
Avg.	0.18	0.13	0.15	0.04	0.23	0.19	0.30	0.12	0.20	0.15

on the scenario and on the experience of the crowd one can argue whether a gold standard generated by a single expert is necessarily more valuable than the group decision of the 6 reviewers. Therefore, one can raise the question whether the response quality is always ”black or white”.

A similar perspective can be taken on the rework process. As mentioned in section 6.1.4, a perfect rework is assumed even for the 89 tasks that were rejected by the sampling plan although they had been answered correctly by the workers according to the gold standard. Although the rework step has not been tested in the scope of this case study, it becomes clear that the rejections are not necessarily an issue. The gold standard does not represent the absolute truth, but is only used as a baseline for the experiments. In practice, there is usually no gold standard available or, if there is, it is subject to the same restrictions. If multiple reviewers agree that there is an issue with a response it might be worth to revise it. Even if it may not be indeed entirely incorrect, taking into account the reviewer's considerations may at least help improve the quality. Altogether, the product research case study confirms that the group validation approach provides a viable mechanisms for managing the quality of non-deterministic cloud labor services.

Given that the majority review approach is an instance of the validation pattern, it is not limited to non-deterministic tasks. According to the decision matrix in section 3.4.1 it can also be applied to deterministic tasks. However, compared to the core model described in chapter 5 the majority review approach can be assumed to be less efficient because it uses a fixed level of redundancy for the inspection steps while the core model dynamically adjusts their redundancy. Apart from that it requires additional rework steps.

As an outlook, the efficiency of the majority review approach could be increased by combining it with the core model for performing the actual reviews. Since the review tasks represent binary decisions, the core model could be applied to them to reduce their redundancy. In a similar way the core model could also be applied to the iteration and the comparison pattern introduced in section 3.3. These considerations further emphasize the strength and the flexibility of the core model.

Part IV.

Conclusion

10. Conclusion

This chapter provides a summary of the thesis, its contributions and limitations and provides recommendations for further research.

After subsuming the thesis and its objectives in section 10.1, section 10.2 summarizes the core contributions being made. Section 10.3 finally addresses limitations of the thesis and provides a number of recommendations for further research.

10.1. Summary

Cloud labor services apply the cloud computing paradigm to human workforce in order to provide work as a highly scalable service that can be utilized by organizations to react more flexibly to peak workloads and varying market demands. A coordination platform acts as an interface between requesters who need to get work done and workers who want to perform work in order to earn money. Because of the lose relation between requesters and workers the primary conceptual challenge of the concept is to deliver high quality work results. This thesis argues that a scalable, efficient and also goal-based quality management (QM) is needed to benefit from the concept to its full extent.

The actual objective has been to investigate how such a QM can be realized. This objective has been approached in multiple steps. As a foundation, the concept of cloud labor services is defined and a comprehensive overview on the state of the art is provided that examines the concept from different perspectives. In a next step, existing QM approaches are discussed and compared to each other and the gaps with regard to the objectives of the thesis are identified. While scalability is not an issue for most of the existing approaches, it turns out that there is actually a lack of efficient and goal-based approaches. In order to close the gap, a new model for QM of cloud labor services is introduced that in particular leverages statistical quality control (SQC). Along with a number of model extensions and variations it was adapted to different types of usage scenarios. The following paragraphs outline the model and its extensions and how they have been validated:

- The core model uses the continuous sampling plan CSP-1 to track the contributions of each worker separately based on samples taken from a stream of work results. A *dynamic voting mechanism* (DVM) is introduced for inspecting the samples. The

DVM requests responses from additional workers and aggregates them by taking into consideration the individual failure rates of the workers. The number of additional responses is dynamically increased until the required inspection quality is reached. Thus, the inspection effort is kept minimal. By also inspecting only a fraction of the tasks, the model is capable of guaranteeing a certain predefined level of result quality at minimum costs. In order to apply the model to research and real world scenarios, an extendable software toolkit has been developed. An evaluation based on Amazon's Mechanical Turk platform has shown a reduction of the QM effort of up to 75% compared to the traditional approaches. A case study performed in cooperation with the commercial cloud labor service platform provider Clickworker.com has proven the robustness of the CSP/DVM and its applicability to business scenarios.

- The *DVM for multi-labeling* represents an extension that makes the DVM applicable to the important class of multi-labeling scenarios. The DVM for multi-labeling was validated in a healthcare business scenario in which a crowd of service center agents is delivering a coding service for medical diagnosis to a private insurance provider. Rather than performing all work manually, the idea is to complement the algorithmic coding solution provided by the Semfinder AG with the manual efforts of the crowd. The concept has been turned into a commercial service offering by IBM that is being used in production by multiple German health insurance providers.
- The *group validation approach* follows a model complementary to the CSP/DVM. It implements the validation rather than the voting pattern which represents another fundamental QM pattern for cloud labor services. Furthermore it does not operate on a continuous stream but on batches of worker responses to which an acceptance sampling plan is applied. Samples are inspected using feedback from multiple reviewers which is consolidated using a maximum likelihood estimation. The approach was successfully evaluated in a third case study in cooperation with the commercial cloud labor platform provider Bitworxx.

10.2. Contribution

The thesis makes contributions on several levels: On the level of the actual research question, on the level of QM for cloud labor services and on the level of the concept of cloud labor services as a whole. The contributions are not limited to the academic body but also entail manifold managerial implications. For each of the levels, the contributions are outlined in the following paragraphs:

Actual research question

This thesis represents the first comprehensive application of SQC to cloud labor services. When applying SQC to cloud labor services, a major challenge is to provide a scalable sample inspection mechanism. The newly developed DVM meets this challenge by providing a well-founded model for dynamic inspection of worker responses. Its combination with the CSP-1 into the CSP/DVM represents the core contribution of the thesis. A profound evaluation based on a model system confirms that the CSP/DVM allows for an efficient, scalable and goal-based QM for cloud labor services. However, even when used standalone, the DVM still delivers a high level of efficiency. As a result, the desired level of complexity can be decided based on the scenario needs. For long running high volume scenarios, the more complex combination with the CSP-1 may be chosen in order to gain higher efficiency while for smaller volumes of tasks, the simpler standalone implementation may be preferred. Both options are supported by the CSP/DVM toolkit. Thanks to a built-in simulation platform which operates on real life results, it allows for repeatable and realistic experiments. Since it is based on the voting pattern which represents the most fundamental pattern for crowd-based QM, the DVM can generally be exploited in practically any type of scenarios. Concrete model extensions and variations are provided for the important category of multi-labeling scenarios as well as for non-deterministic tasks.

The QM approaches developed in this thesis are proven by and adjusted to a number of business scenarios. These scenarios did not only confirm the efficiency, robustness and the reach of the approaches but also helped to identify and to address their limitations. For example, the need for a specific extension for small worker pools (section 5.6) only became clear when applying the mechanism to the Clickworker.com use case. Clear step-by-step instructions facilitate an straightforward implementation of the approaches in real life scenarios for the desired level of complexity. The CSP/DVM toolkit which is available as open source can serve as a template and starting point. Altogether, the thesis confirms that with the DVM and its variations, an efficient, scalable and goal-based QM for cloud labor services can be realized. The case studies further confirm that a goal-based QM is an important tool for enforcing specific quality objectives at the interface between the cloud labor platform and the requester.

QM of cloud labor services

Besides the concrete QM approaches, this thesis contributes to an understanding of the general concept of QM of cloud labor services. Chapter 3 represents the first extensive analysis of QM approaches for cloud labor services. While existing publications typically use the term quality as a synonym for accuracy, a clear and fine granular definition of relevant quality dimensions is provided in this thesis and the factors influencing the quality are identified. Initial considerations about scalability, availability and response time are provided which have not been covered by literature so far. As an important contribution, the existing QM approaches are structured into a systematic classification and the strengths and weaknesses of each approach are identified.

These considerations and specifically the classification of QM approaches are not just of academic interest but also can be considered a useful tool for practitioners. Based on a comprehensible set of task characteristics, a decision matrix and a simplified rule-of-thumb allow for choosing an adequate QM approach for any given scenario.

General concept of cloud labor services

The contributions made by this thesis are not limited to QM. There are only few extensive studies available on cloud labor services. In fact, apart from (Law & Ahn, 2011) which is focusing on the overlapping topic of human computation and (Kittur et al., 2013) which is focusing on the broader topic of crowd work, chapter 2 represents the first comprehensive scientific analysis of the phenomenon of cloud labor services. Besides a definition and an extensive analysis of the research challenges it contributes a detailed and structured presentation of the state of the art covering multiple disciplines, complemented by own contributions. e.g. regarding the relation to cloud computing and the risk of disclosing sensitive data.

This thesis does not only contribute to the academic body of research but it has also gained reasonable awareness in the non-academic world. It exemplifies, how business can benefit from the cloud labor service concept from different perspectives, the requester perspective (Online shopping platform, Telegate, health insurance company), the platform perspective (Bitworxx, Clickworker, IBM), and the workforce perspective (Buw). The thesis also demonstrates how the concept can be utilized to develop new innovative types of business scenarios that open up possibly huge business potentials. The contributions of the thesis have been advertised in a full week demo booth at the German CeBit computer fair (Kern et al., 2010b) and at multiple IBM customer conferences, the IBM IMPACT in Las Vegas (Pfau & Kern, 2011) and the *Versicherungsbetrieb der Zukunft* (VBZ) in Cologne in 2010. Furthermore, non-academic publications about the contributions have been published in multiple non-academic journals (Rademacher, 2011; Satzger & Kern, 2011) and in radio broadcasts on the German radio stations SWR2[1] and Campus Radio. Around the research work, a community of academics and practitioners was built who met at several informal workshops. The *CrowdNet 2012* workshop (Kern et al., 2012c) was attended by 25 academics and 10 practitioners. The *CrowdWork 2013* workshop had been co-located with the *3rd International Conference on Social Computing and its Applications* (SCA) and attracted some 20 participants[2]. One of the focus topics had been ethics in crowdsourcing.

10.3. Recommended further research

The QM approaches introduced in this thesis are subject to a number of specific restrictions which should be further investigated. They are again structured by the three levels used above.

On the level of the research question, it has been made plausible that the DVM can be exploited in other QM patterns like validation, iteration or comparison, but it is not concretely shown how that can be achieved. Specifically, it should be investigated how the efficiency of a dynamic validation approach based on the DVM compares to the group validation approach. Also, while currently being limited to categorical data, the DVM should be extended to non-categorical data in the future.

On the general level of QM for cloud labor services, most of the existing QM approaches are output-based. Alternative ways of QM should be investigated. One possible direction is to further pursue on the recent work regarding execution process monitoring which had been illustrated in section 3.2.3. Another option would be to further investigate the concept of reputation which is mainly used for human-managed cloud labor so far. For cloud labor services, the qualification of workers is typically identified separately for each scenario but no implications are being made between the scenarios. It should be examined whether the capabilities of a worker could be described by a defined set of parameters from which the worker's qualification for a specific scenario could be deduced.

As demonstrated by the medical coding scenario on chapter 9, cloud labor services can be used to complement algorithmic approaches. It should be investigated how an *algorithmic worker* would affect the outcome of the QM patterns. Could one even think of multiple algorithmic workers that together deliver a better result than each of them individually?

While this thesis as well as most of the existing research are examining the *accuracy* of work results delivered by cloud labor services, future research should also address other

[1] `http://mp3-download.swr.de/swr2/impuls/beitraege/2012/04/swr2impuls_20120424_kit_testet_die_people_cloud.6444m.mp3`, last accessed 2013-04-18.
[2] `http://socialcloud.aifb.uni-karlsruhe.de/workshops/CrowdWork/`, last accessed 2013-10-28.

dimensions of quality which have only been briefly discussed in chapter 3. Moreover, the QM mechanisms developed in this thesis have the objective to meet well-defined quality goals in terms of a specific minimum accuracy of the information returned by the cloud labor services no matter what the QM effort in terms of redundancy (and costs) would be. In many business scenarios however, there may rather be a trade-off between quality and costs. As an outlook, appendix B provides preliminary considerations about cost-benefit-objectives for QM.

On the level of cloud labor services in general, the probably most severe limitation of this thesis is that it merely focuses on the conceptual feasibility of cloud labor services and widely neglects their socio-economical implications. One should always remember that at the center of cloud labor services there are humans who are performing the work. Even the best technical platform or the most efficient QM approach is useless if it does not allow for a sustainable work model! Therefore, the socio-economical facets of cloud labor services need to be carefully investigated and discussed, comprising the legal status of the crowd workers and the viability of the work model in general. A broad public discussion is needed to understand and actively shape the role cloud labor services will play in our future working environment.

Bibliography

AAAI. 2013. *Main page of HCOMP - Annual human computation conference, workshop and publication archives.* `http://www.humancomputation.com/`, last accessed on January 21, 2013.

Agrawal, Ashish, Amend, Mike, Das, Manoj, et al. 2007a. *Web services human task (WS-HumanTask), version 1.0.* `http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-bpel4people/WS-HumanTask_v1.pdf`, last accessed on October 25, 2013.

Agrawal, Ashish, Amend, Mike, Das, Manoj, et al. 2007b. *WS-BPEL extension for people.* `http://scn.sap.com/docs/DOC-17718`, last accessed on January 18, 2013.

Alonso, Omar. 2011. Perspectives on infrastructure for crowdsourcing. *Pages 7–10 of: Proceedings of the Workshop on Crowdsourcing for Search and Data Mining (CSDM) at the Fourth ACM International Conference on Web Search and Data Mining (WSDM).*

Alonso, Omar, Rose, Daniel E., and Stewart, Benjamin. 2008. Crowdsourcing for relevance evaluation. *SIGIR Forum*, **42**(2), 9–15.

Amazon Inc. 2013a. *Amazon Mechanical Turk requester website.* `https://requester.mturk.com/`, last accessed on February 6, 2013.

Amazon Inc. 2013b. *Amazon Mechanical Turk worker website FAQs.* `https://www.mturk.com/mturk/help?helpPage=worker`, last accessed on February 6, 2013.

Ambati, Vamshi, Vogel, Stephan, and Carbonell, Jaime G. 2011. Towards task recommendation in micro-task markets. *In: Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence.*

Barr, Jeff, and Cabrera, Luis Felipe. 2006. AI gets a brain. *Queue*, **4**(4), 24–29.

Bauer, Cordula. 2010. *An approach to statistical quality control for human-based electronic services based on multiple reviews per task.* Master thesis, Karlsruhe Institute of Technology.

Berger, Thomas G. 2012. *Service-Level-Agreements: Konzeption und Management von Service-Level-Agreements für IT-Dienstleistungen.* Saarbrücken, Germany: AV Akademikerverlag.

Bermbach, David, Kern, Robert, Wichmann, Pascal, Rath, Sandra, and Zirpins, Christian. 2011. An extendable toolkit for managing quality of human-based electronic services. *In: Proceedings of the 3rd Human Computation Workshop (HCOMP 2011).* AAAI Press.

Bernstein, Michael S., Little, Greg, Miller, Robert C., Hartmann, Björn, Ackerman, Mark S., Karger, David R., Crowell, David, and Panovich, Katrina. 2010. Soylent: A word processor with a crowd inside. *Pages 313–322 of: Proceedings of the 23nd annual ACM symposium on User interface software and technology.* UIST '10. ACM, New York, NY, USA.

Bernstein, Michael S., Brandt, Joel, Miller, Robert C., and Karger, David R. 2011. Crowds in two seconds: Enabling realtime crowd-powered interfaces. *Pages 33–42 of: Proceedings of the 24th annual ACM symposium on User interface software and technology.* UIST '11. New York, NY, USA: ACM.

Biewald, Lukas. 2012. *E-Mail "An Update from CrowdFlower's Founder".*

Bigham, Jeffrey P., Jayant, Chandrika, Ji, Hanjie, Little, Greg, Miller, Andrew, Miller, Robert C., Miller, Robin, Tatarowicz, Aubrey, White, Brandyn, White, Samual, and Yeh, Tom. 2010. VizWiz: Nearly real-time answers to visual questions. *Pages 333–342 of: Proceedings of the 23nd annual ACM symposium on User interface software and technology.* UIST '10. New York, NY, USA: ACM.

Bitworxx GmbH. 2013. *About bitworxx.com.* `http://www.bitworxx.com/cloud-working-crowdsourcing-bitworxx.html`, last accessed on January 29, 2013.

Brabham, Daren C. 2008. Moving the crowd at iStockphoto: The composition of the crowd and motivations for participation in a crowdsourcing application. *First Monday*, **13**(6), 1–22.

Buw Group. 2013. *Homepage of buw customer care operations and consulting.* `http://www.buw.de/en/startseite.html`, last accessed on February 4, 2013.

Callison-Burch, Chris. 2009. Fast, cheap, and creative: Evaluating translation quality using Amazon's Mechanical Turk. *Pages 286–295 of: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing.* EMNLP '09. Stroudsburg, PA, USA: Association for Computational Linguistics.

Chesbrough, Henry William. 2003. *Open innovation: The new imperative for creating and profiting from technology.* Boston, MA, USA: Harvard Business Review Press.

Chilton, Lydia B., Horton, John J., Miller, Robert C., and Azenkot, Shiri. 2010. Task search in a human computation market. *Pages 1–9 of: Proceedings of the ACM SIGKDD Workshop on Human Computation.* HCOMP 2010. New York, NY, USA: ACM.

Clariana, Roy B., Wagner, Daren, and Roher Murphy, Lucia C. 2000. Applying a connectionist description of feedback timing. *Educational Technology Research and Development*, **48**(3), 5–22.

Corney, Jonathan R., Torres-Sánchez, Carmen, Jagadeesan, Amanda Prasanna, Yan, X. T., Regli, William C., and Medellin, Hugo. 2010. Putting the crowd to work in a knowledge-based factory. *Advanced Engineering Informatics*, **24**(3), 243–250.

Cosley, Dan, Frankowski, Dan, Terveen, Loren, and Riedl, John. 2007. SuggestBot: Using intelligent task routing to help people find work in wikipedia. *Pages 32–41 of: Proceedings of the 12th international conference on Intelligent user interfaces.* ACM.

Crowdflower. 2010. *List of sponsors of CrowdConf 2011 crowdsourcing conference in San Francisco, CA, USA.* `http://www.crowdconf2011.com/sponsors.html`, last accessed on April 6, 2012.

CrowdSource. *Scalable Workforce LLC announces the acquisition of CrowdSource.com.* `http://www.crowdsource.com/about-us/press/scalable-workforce-llc-announces-the-acquisition-of-crowdsource-com/`, last accessed on June 4, 2012.

Crowdsourcing LLC. 2012. *Crowdsourcing Industry Report.* `http://www.crowdsourcing.org/editorial/enterprise-crowdsourcing-research-report-by-massolution/11736`, last accessed on October 25, 2013.

Crowdsourcing.org. 2012a. *Crowdsourcing and crowdfunding - The industry website.* `http://www.crowdsourcing.org/`, last accessed on May 31, 2012.

Crowdsourcing.org. 2012b. *Directory of crowdsourcing sites.* `http://www.crowdsourcing.org/directory`, last accessed on April 6, 2012.

Cuel, Roberta, Kern, Robert, Schulze, Thimo, and Krause, Markus. 2013. *Website of 1st International Workshop on Crowd Work and Human Computation.* `http://socialcloud.aifb.uni-karlsruhe.de/workshops/CrowdWork/`, last accessed on October 28, 2013.

Dai, Peng, Mausam, and Weld, Daniel S. 2010. Decision-theoretic control of crowd-sourced workflows. *In: Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI 2010).*

Dai, Peng, Mausam, and Weld, Daniel S. 2011. Artificial intelligence for artificial artificial intelligence. *In: Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence (AAAI 2011).*

Dawid, Alexander P., and Skene, Allan M. 1979. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Journal of the Royal Statistical Society*, **28**(1), 20–28.

de Caritat marquis de Condorcet, Jean-Antoine-Nicolas. 1785. *Essai sur l'application de l'analyse a la probabilite des decisions rendues a la pluralite des voix.* 1st edn. Paris, France: De l'Imprimerie royale.

De Feo, Joseph, and Juran, Joseph M. 2010. *Juran's quality handbook: The complete guide to performance excellence.* 6th edn. New York, NY, USA: McGraw-Hill Professional.

Deci, Edward L., and Ryan, Richard M. 1985. *Intrinsic motivation and self-determination in human behavior.* New York, NY, USA: Plenum Press.

Deming, William Edwards. 2000. *Out of the crisis.* 1st edn. Cambridge, MA, USA: MIT Press.

Dempster, Arthur P., Laird, Nan M., and Rubin, Donald B. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, **39**(1), 1–38.

Deng, Jia, Dong, Wei, Socher, R, Li, Li-Jia, Li, Kai, and Fei-fei, Li. 2009. ImageNet: A large-scale hierarchical image database. *Pages 248–255 of: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on.*

DiPalantino, Dominic, Karagiannis, Thomas, and Vojnovic, Milan. 2011. *Individual and collective user behavior in crowdsourcing services.* Tech. rept. Technical report, Microsoft Research.

Dodge, Harold F. 1943. A sampling inspection plan for continuous production. *Annals of Mathematical Statistics*, **14**(3), 264–279.

Dodge, Harold F., and Romig, Harry G. 1959. *Sampling inspection tables, single and double sampling.* 2nd edn. New York, NY, USA: John Wiley.

Dodge, Harold F., and Torrey, Mary N. 1951. Additional continuous sampling inspection plans. *Industrial Quality Control*, **7**(5), 7–12.

Eckert, Kai, Niepert, Mathias, Niemann, Christof, Buckner, Cameron, Allen, Colin, and Stuckenschmidt, Heiner. 2010. Crowdsourcing the assembly of concept hierarchies. *Pages 139–148 of: Proceedings of the 10th annual joint conference on Digital libraries.* JCDL '10. New York, NY, USA: ACM.

Faridani, Siamak, Hartmann, Björn, and Ipeirotis, Panagiotis G. 2011. What's the right price? pricing tasks for finishing on time. *In: Proceedings of the 3rd Human Computation Workshop (HCOMP 2011).* AAAI Press.

Felstiner, Alek. 2011. Employment and labor law in the crowdsourcing industry. *Berkeley Journal of Employment and Labor Law*, **32**(1), 143–202.

Frei, Brent. 2009 (Sept.). *Paid Crowdsourcing.* Tech. rept. Smartsheet.com.

Gentry, Craig. 2009. Secure distributed human computation. *Pages 181–189 of:* Christianson, Bruce, Crispo, Bruno, Malcolm, JamesA., and Roe, Michael (eds), *Security Protocols.* Lecture Notes in Computer Science, vol. 5087. Berlin/Heidelberg, Germany: Springer.

Google Inc. 2011. *Official announcement of Google Labs shutdown.* `http://googleblog.blogspot.co.uk/2011/07/more-wood-behind-fewer-arrows.html`, last accessed on February 2, 2013.

Gordon, Jonathan, Van Durme, Benjamin, and Schubert, Lenhart K. 2010. Evaluation of commonsense knowledge with Mechanical Turk. *Pages 159–162 of: Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk.* CSLDAMT '10. Stroudsburg, PA, USA: Association for Computational Linguistics.

Gosh, D. T. 1996. An optimum continuous sampling plan CSP-2 with k $\neq$ i to minimise the amount of inspection when incoming quality p follows a distribution. *The Indian Journal of Statistics*, **58**(1), 105–117.

Grady, Catherine, and Lease, Matthew. 2010. Crowdsourcing document relevance assessment with Mechanical Turk. *Pages 172–179 of: Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk.* Association for Computational Linguistics, Stroudsburg, PA, USA.

Gräfen, Manfred. 2012. *Automation in der Versicherungswirtschaft, Vortrag bei der Veranstaltung "IT-Trends in der Versicherungswirtschaft" der Gesellschaft für Informatik.* `http://rg-koeln.gi.de/uploads/media/Vortrag_Graefen_01.pdf`, last accessed on July 30, 2013.

Graham, Ronald L., Knuth, Donald E., and Patashnik, Oren. 1994. *Concrete mathematics: A foundation for computer science.* 2nd edn. Amsterdam, Netherlands: Addison-Wesley Professional.

Grier, David Alan. 2007. *When computers were human.* Princeton, NJ, USA: Princeton University Press.

Grier, David Alan. 2010. *Let us now compute: The lessons of crowdsourcing.* Recording of presentation at CrowdConf 2010. `http://www.crowdconf2010.com/computershuman.html`, last accessed on October 25, 2013.

GS1. 2006. *An Introduction to the Global Trade Item Number (GTIN).* Tech. rept. GS1, New Jersey, USA.

Hattie, John, and Timperley, Helen. 2007. The power of feedback. *Review of Educational Research*, **77**(1), 81–112.

Heilman, Michael, and Smith, Noah A. 2010. Rating computer-generated questions with Mechanical Turk. *Pages 35–40 of: Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk.* Association for Computational Linguistics, Stroudsburg, PA, USA.

Hermes, Heinz-Josef, and Schwarz, Gerd. 2005. *Outsourcing: Chancen und Risiken, Erfolgsfaktoren, rechtssichere Umsetzung.* München, Germany: Haufe-Lexware.

Hirth, Matthias., Hoßfeld, Tobias, and Tran-Gia, Phuoc. 2011. Anatomy of a crowdsourcing platform-using the example of microworkers.com. *Pages 322–329 of: Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS).* IEEE.

Hirth, Matthias, Hoßfeld, Tobias, and Tran-Gia, Phuoc. 2013. Analyzing costs and accuracy of validation mechanisms for crowdsourcing platforms. *Mathematical and Computer Modelling*, **57**, 2918–2932.

Hoffmann, Leah. 2009. Crowd control. *Communications of the ACM*, **52**(3), 16–17.

Horton, John Joseph, and Chilton, Lydia B. 2010. The labor economics of paid crowdsourcing. *Pages 209–218 of: Proceedings of the 11th ACM conference on Electronic Commerce.* EC '10. Cambridge, MA, USA: ACM.

Howe, Jeff. 2006a. The rise of crowdsourcing. *Wired Magazine*, **14**(6), 1–4.

Howe, Jeff. 2006b. *Crowdsourcing: A Definition.* `http://crowdsourcing.typepad.com/cs/2006/06/crowdsourcing_a.html`, last accessed on January 18, 2013.

Hsueh, Pei-Yun, Melville, Prem, and Sindhwani, Vikas. 2009. Data quality from crowdsourcing: a study of annotation selection criteria. *Pages 27–35 of: Proceedings of the NAACL HLT 2009 Workshop on Active Learning for Natural Language Processing.* Association for Computational Linguistics, Stroudsburg, PA, USA.

Huang, Eric, Zhang, Haoqi, Parkes, David C., Gajos, Krzysztof Z., and Chen, Yiling. 2010. Toward automatic task design: A progress report. *Pages 77–85 of: Proceedings of the ACM SIGKDD Workshop on Human Computation.* HCOMP 2010. New York, NY, USA: ACM.

Humangrid GmbH. 2013. *About the Clickworker platform.* `http://www.clickworker.com/en/about-us/`, last accessed on January 18, 2013.

IBM. 2008. *IBM in Deutschland - Das Unternehmen - Branchen - Insurance - Deutschland.* `http://www-05.ibm.com/de/ibm/leistungen/insurance.html`, last accessed on February 3, 2013.

IBM. 2012. *IBM Deutschland Research & Development GmbH - Company profile.* `http://www-05.ibm.com/de/entwicklung/ueberuns/index_en.html`, last accessed on February 4, 2013.

IG Metall. 2013. *Crowdsourcing: Beschäftigte im globalen Wettbewerb um Arbeit - am Beispiel IBM.* Tech. rept. IG Metall, Frankfurt, Germany.

International Data Corporation. 2013. *Worldwide and U.S. Business Process Outsourcing Services 2013-2017 Forecast: Time for Customers to Reap the Emerging Benefits of Transformative and Agile BPO Services.* Tech. rept. International Data Corporation (IDC), Framingham, MA, USA.

Ipeirotis, Panagiotis G. 2010a. Analyzing the Amazon Mechanical Turk marketplace. *XRDS*, **17**(2), 16–21.

Ipeirotis, Panagiotis G., Provost, Foster, and Wang, Jing. 2010. Quality management on Amazon Mechanical Turk. *Pages 64–67 of: Proceedings of the ACM SIGKDD Workshop on Human Computation.* HCOMP 2010. New York, NY, USA: ACM.

Ipeirotis, Panos. 2008. *Mechanical Turk: The demographics | A Computer Scientist in a Business School.* `http://behind-the-enemy-lines.blogspot.com/2008/03/mechanical-turk-demographics.html`, last accessed on October 30, 2012.

Ipeirotis, Panos. 2010b. *Demographics of Mechanical Turk.* Working paper. `http://ssrn.com/abstract=1585030`, last accessed on May 8, 2013.

Ipeirotis, Panos. 2012. *Discussion on Disintermediating a Labor Channel | A Computer Scientist in a Business School.* `http://www.behind-the-enemy-lines.com/2012/07/discussion-on-disintermediating-labor.html`, last accessed on May 1, 2013.

ISO - The International Organization for Standardization. 2005. *ISO 9000:2005 - Quality Management systems - Fundamentals and vocabulary.*

Kahn, Beverly K., Strong, Diane M., and Wang, Richard Y. 2002. Information quality benchmarks: Product and service performance. *Communications of the ACM*, **45**(4), 184–192.

Kaisser, Michael, and Lowe, John. 2008. Creating a research collection of question answer sentence pairs with Amazon's Mechanical Turk. *In: Proceedings of the Sixth International Language Resources and Evaluation (LREC'08).*

Kaufmann, Nicolas, Schulze, Thimo, and Veit, Daniel. 2011. More than fun and money. Worker motivation in crowdsourcing - A study on Mechanical Turk. *In: Proceedings of the 17th Americas Conference on Information Systems (AMCIS 2011).*

Kern, Robert, Zirpins, Christian, and Agarwal, Sudhir. 2009a. Managing Quality of Human-Based eServices. *Pages 304–309 of:* Feuerlicht, George, and Lamersdorf, Winfried (eds), *Service-Oriented Computing - ICSOC 2008 Workshops, ICSOC 2008 International Workshops, Sydney, Australia, December 1st, 2008, Revised Selected Papers.* Lecture Notes in Computer Science, vol. 5472. Springer.

Kern, Robert, Zirpins, Christian, and Satzger, Gerhard. 2009b. *People Services: Manpower as a Service.* Presentation at the 18th Annual Frontiers in Service Conference, Honolulu, USA.

Kern, Robert, Bauer, Cordula, Thies, Hannes, and Satzger, Gerhard. 2010a. *Efficient quality management for human-based electronic services.* Presentation at the 19th Annual Frontiers in Service Conference, Karlstad, Sweden.

Kern, Robert, Peisl, Roland, Uhlenberg, David, Pasing, Andreas, and Führich, Werner. 2010b. PeopleClouds: Scalable workforce as-a-service. Hannover, Germany: Demo, IBM booth at CeBIT trade show.

Kern, Robert, Thies, Hans, Bauer, Cordula, and Satzger, Gerhard. 2010c. Quality assurance for human-based electronic services: A decision matrix for choosing the right approach. *Pages 421–424 of:* Daniel, Florian, and Facca, Federico (eds), *Current Trends in Web Engineering.* Lecture Notes in Computer Science, vol. 6385. Berlin/Heidelberg, Germany: Springer.

Kern, Robert, Thies, Hans, and Satzger, Gerhard. 2010d. Statistical quality control for human-based electronic services. *Pages 243–257 of:* Maglio, Paul P., Weske, Mathias, Yang, Jian, and Fantinato, Marcelo (eds), *Service-Oriented Computing - 8th International Conference, ICSOC 2010, San Francisco, CA, USA, December 7-10, 2010. Proceedings.* Lecture Notes in Computer Science, vol. 6470. Springer.

Kern, Robert, Bauer, Cordula, Thies, Hannes, and Satzger, Gerhard. 2010e. Validating results of human-based electronic services leveraging multiple reviewers. *In: Proceedings of the 16th Americas Conference on Information Systems (AMCIS 2010).*

Kern, Robert, Thies, Hannes, and Satzger, Gerhard. 2011. Efficient quality management of human-based electronic services leveraging group decision making. *In: Proceedings of the 19th European Conference on Information Systems (ECIS 2011).*

Kern, Robert, Bermbach, David, Rath, Sandra, Wichmann, Pascal, and Meller, Jan. 2012a. *cspwmw - Toolkit for dynamic statistical quality management on human computation or crowdsourcing platforms - Google Project Hosting.* `http://code.google.com/p/cspwmw/`, last accessed on April 5, 2013.

Kern, Robert, Thies, Hans, Zirpins, Christian, and Satzger, Gerhard. 2012b. Dynamic and goal-based quality management for human-based electronic services. *International Journal of Cooperative Information Systems*, **21**(1), 3–29.

Kern, Robert, Schulze, Thimo, Krause, Markus, and Cuel, Roberta. 2012c. *Website of CrowdNet workshops on cloud labor and human computation.* `http://www.ksri.kit.edu/crowdnet/`, last accessed on January 21, 2013.

Kieninger, Axel, Satzger, Gerhard, Straeten, Detlef, Schmitz, Björn, and Baltadzhiev, Dian. 2012. Business Cost Budgets - A Methodology to Incorporate Business Impact into Service Level Agreements. *International Journal of Service Science, Management, Engineering, and Technology (IJSSMET)*, **3**(3), 49–64.

Kittur, Aniket. 2010. Crowdsourcing, collaboration and creativity. *XRDS*, **17**(2), 22–26.

Kittur, Aniket, Chi, Ed H., and Suh, Bongwon. 2008. Crowdsourcing user studies with Mechanical Turk. *Pages 453–456 of: Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems.* New York, NY, USA: ACM.

Kittur, Aniket, Smus, Boris, Khamkar, Susheel, and Kraut, Robert E. 2011. Crowdforge: Crowdsourcing complex work. *Pages 43–52 of: Proceedings of the 24th annual ACM symposium on User interface software and technology.* UIST '11. New York, NY, USA: ACM.

Kittur, Aniket, Khamkar, Susheel, André, Paul, and Kraut, Robert. 2012. CrowdWeaver: visually managing complex crowd work. *Pages 1033–1036 of: Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work.* ACM.

Kittur, Aniket, Nickerson, Jeffrey V., Bernstein, Michael, Gerber, Elizabeth, Shaw, Aaron, Zimmerman, John, Lease, Matt, and Horton, John. 2013. The future of crowd work. *Pages 1301–1318 of: Proceedings of the 2013 conference on Computer supported cooperative work.* CSCW '13. New York, NY, USA: ACM.

Kleemann, Frank, Voß, G., and Rieder, Kerstin. 2008. Un(der)paid innovators: The commercial utilization of consumer work through crowdsourcing. *Science, Technology & Innovation Studies*, **4**(1).

Kolodny, Lora. 2012. *The founders: Liquidspace CEO on mission to find you a workspace - Venture Capital Dispatch - WSJ.* `http://blogs.wsj.com/venturecapital/2012/05/31/the-founders-liquidspace-ceo-on-mission-to-find-you-a-workspace/`, last accessed on April 3, 2013.

Kulkarni, Anand, Can, Matthew, and Hartmann, Björn. 2012. Collaboratively crowdsourcing workflows with turkomatic. *Pages 1003–1012 of: Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work.* CSCW '12. New York, NY, USA: ACM.

Kulkarni, Anand P., Can, Matthew, and Hartmann, Björn. 2011. Turkomatic: Automatic recursive task and workflow design for Mechanical Turk. *Pages 2053–2058 of: CHI '11 Extended Abstracts on Human Factors in Computing Systems.* CHI EA '11. ACM, New York, NY, USA.

Lakhani, Karim R., Jeppesen, Lars B., Lohse, Peter A., and Panetta, Jill A. 2007. *The value of openess in scientific problem solving.* Boston, MA, USA: Division of Research, Harvard Business School.

Lasecki, Walter S., Murray, Kyle I., White, Samuel, Miller, Robert C., and Bigham, Jeffrey P. 2011. Real-time crowd control of existing interfaces. *Pages 23–32 of: Proceedings of the 24th annual ACM symposium on User interface software and technology.* UIST '11. New York, NY, USA: ACM.

Law, Edith, and Ahn, Luis. 2011. Human computation. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, **5**(3), 1–121.

Le, John, Edmonds, Andy, Hester, Vaughn, and Biewald, Lukas. 2010. Ensuring quality in crowdsourced search relevance evaluation: The effects of training question distribution. *Pages 21–26 of: Proceedings of the ACM SIGIR 2010 workshop on crowdsourcing for search evaluation (CSE 2010).*

Lévy, Pierre. 2001. Collective intelligence. *Reading digital culture*, **4**, 253–258.

Lieberman, Gerald J., and Solomon, Herbert. 1955. Multi-level continuous sampling plans. *The Annals of Mathematical Statistics*, **26**(4), 686–704.

Liem, Beatrice, Zhang, Haoqi, and Chen, Yiling. 2011. An iterative dual pathway structure for speech-to-text transcription. *In: Proceedings of the 3rd Human Computation Workshop (HCOMP 2011)*, vol. 2011. AAAI Press.

Little, Greg, Chilton, Lydia B., Goldman, Max, and Miller, Rob. 2009a. TurKit: Tools for iterative tasks on Mechanical Turk. *Pages 29–30 of: Proceedings of the ACM SIGKDD Workshop on Human Computation.* HCOMP 2009.

Little, Greg, Chilton, Lydia B., Goldman, Max, and Miller, Robert C. 2009b. Turkit: Tools for iterative tasks on Mechanical Turk. *Pages 29–30 of: Proceedings of the ACM SIGKDD Workshop on Human Computation.* HCOMP 2009. New York, NY, USA: ACM.

Little, Greg, Chilton, Lydia B., Goldman, Max, and Miller, Robert C. 2010a. Exploring iterative and parallel human computation processes. *Pages 68–76 of: Proceedings of the ACM SIGKDD workshop on human computation.* HCOMP 2010. ACM, New York, NY, USA.

Little, Greg, Chilton, Lydia B., Goldman, Max, and Miller, Robert C. 2010b. Turkit: Human computation algorithms on Mechanical Turk. *Pages 57–66 of: Proceedings of the 23nd annual ACM symposium on User interface software and technology.* UIST '10. ACM, New York, NY, USA.

Lloyd, Vernon, and Rudd, Colin. 2007. *ITIL Service Design.* London, UK: The Stationery Office.

Lopresti, Daniel. 2009. Optical character recognition errors and their effects on natural language processing. *International Journal on Document Analysis and Recognition (IJDAR)*, **12**(3), 141–151.

Madnani, Nitin, Boyd-Graber, Jordan, and Resnik, Philip. 2010. Measuring transitivity using untrained annotators. *Pages 188–194 of: Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk.* CSLDAMT '10. Stroudsburg, PA, USA: Association for Computational Linguistics.

Malone, Thomas W., Laubacher, Robert, and Dellarocas, Chrysanthos. 2009. Harnessing crowds: Mapping the genome of collective intelligence. *MIT Sloan Research Paper.*

Marge, Matthew, Banerjee, Satanjeev, and Rudnicky, Alexander I. 2010. Using the Amazon Mechanical Turk for transcription of spoken language. *Pages 5270–5273 of: Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on.*

Marschak, Jacob, and Radner, Roy. 1958. *Economic Theory of Teams. Chapter 1.* Cowles Foundation Discussion Papers 59a. Cowles Foundation for Research in Economics, Yale University.

Mason, Winter, and Suri, Siddharth. 2011. Conducting behavioral research on Amazon's Mechanical Turk. *Behavior research methods*, **44**(1), 1–23.

Mason, Winter, and Watts, Duncan J. 2010. Financial incentives and the "performance of crowds". *ACM SIGKDD Explorations Newsletter*, **11**(2), 100–108.

Mason, Winter, and Watts, Duncan J. 2012. Collaborative learning in networks. *Proceedings of the National Academy of Sciences*, **109**(3), 764–769.

McCarthy, John. 2007. *What is artificial intelligence?* `http://www-formal.stanford.edu/jmc/whatisai/whatisai.html`, last accessed on April 4, 2013.

Mell, Peter, and Grance, Timothy. 2011. The NIST definition of cloud computing (draft). *NIST Special Publication*, **800**(145).

Mellebeek, Bart, Benavent, Francesc, Grivolla, Jens, Codina, Joan, Costa-jussà, Marta R., and Banchs, Rafael. 2010. Opinion mining of spanish customer comments with non-expert annotations on Mechanical Turk. *Pages 114–121 of: Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk.* CSLDAMT '10. Association for Computational Linguistics, Stroudsburg, PA, USA.

Meller, Jan. 2012. *Robust, efficient and scalable quality management for paid crowdsourcing platforms.* Bachelor thesis, Karlsruhe Institute of Technology.

Microtask Oy. *About Microtask: Our work so far.* `http://www.microtask.com/cases`, last accessed on February 6, 2013.

Minder, Patrick, and Bernstein, Abraham. 2011. CrowdLang: First steps towards programmable human computers for general computation. *In: Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence.*

Montgomery, Douglas C. 2008. *Introduction to statistical quality control.* 6th edn. New York, NY, USA: John Wiley and Sons.

Negri, Matteo, and Mehdad, Yashar. 2010. Creating a bi-lingual entailment corpus through translations with Mechanical Turk: $100 for a 10-day rush. *Pages 212–216 of: Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk.* Association for Computational Linguistics, Stroudsburg, PA, USA.

Novotney, Scott, and Callison-Burch, Chris. 2010a. Cheap, fast and good enough: automatic speech recognition with non-expert transcription. *Pages 207–215 of: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics.* HLT '10. Stroudsburg, PA, USA: Association for Computational Linguistics.

Novotney, Scott, and Callison-Burch, Chris. 2010b. Shared task: Crowdsourced accessibility elicitation of Wikipedia articles. *Pages 41–44 of: Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk.* CSLDAMT '10. Stroudsburg, PA, USA: Association for Computational Linguistics.

OASIS. 2006. *Reference model for service oriented architecture 1.0.* http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf, last accessed on January 18, 2013.

Ohno, Taiichi. 1988. *Toyota production system: Beyond large-scale production.* 1st edn. New York, NY, USA: Productivity Press.

Oleson, David, Sorokin, Alexander, Laughlin, Greg, Hester, Vaughn, Le, John, and Biewald, Lukas. 2011. Programmatic gold: Targeted and scalable quality assurance in crowdsourcing. *In: Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence.*

Overby, Stephanie. *Outsourcing Definition and Solutions.* http://www.cio.com/article/40380/Outsourcing_Definition_and_Solutions, last accessed on January 18, 2013.

Pande, Peter S., Neuman, Robert P., and Cavanagh, Roland R. 2000. *The six sigma way.* 1st edn. New York, NY, USA: McGraw-Hill.

Parameswaran, Manoj, and Whinston, Andrew B. 2007. Social computing: An overview. *Communications of the Association for Information Systems*, **19**(37), 762–780.

Parasuraman, A., Zeithaml, Valarie A., and Berry, Leonard L. 1985. A conceptual model of service quality and its implications for future research. *The Journal of Marketing*, **49**, 41–50.

Parent, Gabriel, and Eskenazi, Maxine. 2010. Clustering dictionary definitions using Amazon Mechanical Turk. *Pages 21–29 of: Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk.* Association for Computational Linguistics, Stroudsburg, PA, USA.

Parent, Gabriel, and Eskenazi, Maxine. 2011. Speaking to the Crowd: looking at past achievements in using crowdsourcing for speech and predicting future challenges. *Pages 3037–3040 of: Proceedings of Interspeech.*

Paton, Scott M. 2002. *Juran: A lifetime of quality.* http://www.qualitydigest.com/aug02/articles/01_article.shtml, last accessed on November 1, 2012.

Pfau, Gerhard, and Kern, Robert. 2011. People Services: Efficient work processes leveraging the power of the crowd. Las Vegas, NV, USA: Presentation at IBM IMPACT global conference.

PMI. 2004. *A Guide to the Project Management Body of Knowledge (Pmbok Guide).* 3rd edn. Newtown Square, PA, USA: Project Management Institute.

Quinn, Alexander J., and Bederson, Benjamin B. 2011. Human computation: A survey and taxonomy of a growing field. *Pages 1403–1412 of: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.* CHI '11. ACM, New York, NY, USA.

Rademacher, Rochus. 2011. Arbeitskraft in der Wolke. *Digital - Die Zeitschrift für die Informationsgesellschaft*, July, 26–27.

Rashtchian, Cyrus, Young, Peter, Hodosh, Micah, and Hockenmaier, Julia. 2010. Collecting image annotations using Amazon's Mechanical Turk. *Pages 139–147 of: Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk.* CSLDAMT '10. Stroudsburg, PA, USA: Association for Computational Linguistics.

Raykar, Vikas C., Yu, Shipeng, Zhao, Linda H., Jerebko, Anna, Florin, Charles, Valadez, Gerardo Hermosillo, Bogoni, Luca, and Moy, Linda. 2009. Supervised learning from multiple experts: Whom to trust when everyone lies a bit. *Pages 889–896 of: Proceedings of the 26th Annual International Conference on Machine Learning.* ICML '09. New York, NY, USA: ACM.

Raykar, Vikas C., Yu, Shipeng, Zhao, Linda H., Valadez, Gerardo Hermosillo, Florin, Charles, Bogoni, Luca, and Moy, Linda. 2010. Learning from crowds. *Journal of Machine Learning Research*, **99**, 1297–1322.

Reddy, Sasank, Estrin, Deborah, and Srivastava, Mani. 2010. Recruitment framework for participatory sensing data collections. *Pages 138–155 of:* Floreen, Patrik, Krüger, Antonio, and Spasojevic, Mirjana (eds), *Pervasive Computing.* Lecture Notes in Computer Science, vol. 6030. Berlin/Heidelberg, Germany: Springer.

Rinne, Horst, and Mittag, Hans-Joachim. 1995. *Statistische Methoden der Qualitätssicherung.* 3rd edn. München, Germany: Carl Hanser Verlag.

Rogstadius, Jakob, Kostakos, Vassilis, Kittur, Aniket, Smus, Boris, Laredo, Jim, and Vukovic, Maja. 2011. An assessment of intrinsic and extrinsic motivation on task performance in crowdsourcing markets. *In: Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media.*

Rose, Kenneth. 2005. *Project quality management: Why, what and how.* Boca Raton, FL, USA: J. Ross Publishing.

Ross, Joel, Irani, Lilly, Silberman, M. Six, Zaldivar, Andrew, and Tomlinson, Bill. 2010. Who are the crowdworkers?: Shifting demographics in Mechanical Turk. *Pages 2863–2872 of: CHI '10 Extended Abstracts on Human Factors in Computing Systems.* CHI EA '10. New York, NY, USA: ACM.

Rzeszotarski, Jeffrey, and Kittur, Aniket. 2012. CrowdScape: Interactively visualizing user behavior and output. *Pages 55–62 of: Proceedings of the 25th annual ACM symposium on User interface software and technology.* UIST '12. New York, NY, USA: ACM.

Rzeszotarski, Jeffrey M., and Kittur, Aniket. 2011. Instrumenting the crowd: Using implicit behavioral measures to predict task performance. *Pages 13–22 of: Proceedings of the 24th annual ACM symposium on User interface software and technology.* UIST '11. New York, NY, USA: ACM.

Sasaki, Yutaka, Rea, Brian, and Ananiadou, Sophia. 2007. Multi-topic aspects in clinical text classification. *Pages 62–70 of: Bioinformatics and Biomedicine (BIBM), 2007 IEEE International Conference on.*

Satzger, Benjamin, Psaier, Harald, Schall, Daniel, and Dustdar, Schahram. 2011a. Stimulating Skill Evolution in Market-Based Crowdsourcing. *Pages 66–82 of:* Rinderle-Ma, Stefanie, Toumani, Farouk, and Wolf, Karsten (eds), *Business Process Management.* Lecture Notes in Computer Science, vol. 6896. Springer.

Satzger, Gerhard, and Kern, Robert. 2011. People-Cloud: Skalierbare Arbeitskraft aus der Wolke. *Funkschau*, Sept., 18–19.

Satzger, Gerhard, Kern, Robert, Pasing, Andreas, Pfau, Gerhard, Fritz, Ronald, Alexander, Schmid, and Roland, Peisl. 2011b. *People Services: Efficient work processes leveraging the power of the crowd.* Presentation at the 20th Annual Frontiers in Service Conference, Columbus, OH, USA.

Schall, Daniel. 2011. A human-centric runtime framework for mixed service-oriented systems. *Distributed and Parallel Databases*, **29**, 333–360. 10.1007/s10619-011-7081-z.

Scheer, August-Wilhelm, and Brabänder, Eric. 2010. The process of business process management. *Handbook on Business Process Management 2*, 239–265.

Schulze, Thimo, Seedorf, Stefan, Geiger, David, Kaufmann, Nicolas, and Schader, Martin. 2011. Exploring task properties in crowdsourcing - An empirical study on Mechanical Turk. *Page Paper 122 of: Proceedings of the 19th European Conference on Information Systems (ECIS 2011).*

Semfinder AG. 2013. *Semfinder: Company portrait.* `http://www.semfinder.com/en/the-company/the-company/portrait.html`, last accessed on February 4, 2013.

Shahaf, Dafna, and Amir, Eyal. 2007. Towards a theory of AI completeness. *In: Proceedings of 8th Interational symposium on logic formalizations of commonsense reasoning.*

Sheng, Victor, Provost, Foster, and Ipeirotis, Panagiotis G. 2008. Get another label? Improving data quality and data mining using multiple, noisy labelers. *Pages 614–622 of: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining KDD 08.* New York, NY, USA: ACM Press.

Silberman, M. Six, Irani, Lilly, and Ross, Joel. 2010. Ethics and tactics of professional crowdwork. *XRDS*, **17**(2), 39–43.

Snow, Rion, O'Connor, Brendan, Jurafsky, Daniel, and Ng, Andrew Y. 2008. Cheap and fast—but is it good? Evaluating non-expert annotations for natural language tasks. *Pages 254–263 of: Proceedings of the Conference on Empirical Methods in Natural Language Processing.* EMNLP '08. Stroudsburg, PA, USA: Association for Computational Linguistics.

Sorokin, Alexander, and Forsyth, David. 2008. Utility data annotation with Amazon Mechanical Turk. *Pages 1–8 of: Proceedings of the First IEEE Workshop on Internet Vision at CVPR 08.* Washington, WA, USA: IEEE.

Sturm, Rick, Morris, Wayne, and Jander, Mary. 2000. *Foundations of service level management.* Indianapolis, IN, USA: Sams Publishing.

Suri, Siddharth, Goldstein, Daniel G., and Mason, Winter A. 2011. Honesty in an online labor market. *In: Proceedings of the 3rd Human Computation Workshop (HCOMP 2011).* AAAI Press.

Surowiecki, James. 2004. *The wisdom of crowds.* 1st edn. New York, NY, USA: Doubleday.

Thies, Hans. 2010. *Efficient quality management of human-based electronic services leveraging redundant task execution.* Diploma thesis, Karlsruhe Institute of Technology.

Tsoumakas, Grigorios, and Katakis, Ioannis. 2007. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining (IJDWM)*, **3**(3), 1–13.

Turian, Joseph. 2012. *Sector RoadMap: crowd labor platforms in 2012.* Tech. rept. Giga Omni Media, Inc.

Vargo, Stephen L., and Lusch, Robert F. 2004. Evolving to a new dominant logic for marketing. *Journal of Marketing*, **68**(1), 1–17.

von Ahn, Luis. 2005. *Human computation.* PhD Thesis, Carnegie Mellon University, Pittsburgh, PA, USA.

von Ahn, Luis, and Dabbish, Laura. 2004. Labeling images with a computer game. *Pages 319–326 of: Proceedings of the 2004 conference on Human factors in computing systems - CHI '04.* New York, New York, USA: ACM Press.

von Ahn, Luis, and Dabbish, Laura. 2008. Designing games with a purpose. *Communications of the ACM*, **51**(8), 58–67.

Vukovic, Maja, Laredo, Jim, and Rajagopal, Sriram. 2010. Challenges and Experiences in Deploying Enterprise Crowdsourcing Service. *Pages 460–467 of:* Benatallah, Boualem, Casati, Fabio, Kappel, Gerti, and Rossi, Gustavo (eds), *Web Engineering.* Lecture Notes in Computer Science, vol. 6189. Springer Berlin Heidelberg.

Wang, Reay-Chen, and Chen, Chung-Ho. 1997. Minimum average fraction inspected for continuous sampling plan CSP-1 under inspection error. *Journal of Applied Statistics*, **24**(5), 539–548.

Warfield, Simon K., Zou, Kelly H., and Wells, William M. 2004. Simultaneous truth and performance level estimation (STAPLE): an algorithm for the validation of image segmentation. *IEEE Transactions on Medical Imaging*, **23**(7), 903–921.

Weld, Daniel S., Mausam, and Dai, Peng. 2011. Human intelligence needs artificial intelligence. *In: Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence.*

Welinder, Peter, and Perona, Pietro. 2010. Online crowdsourcing: Rating annotators and obtaining cost-effective labels. *Pages 25–32 of: Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on.* IEEE Computer Society.

White, Samuel. 2010. Audiowiz: Nearly real-time audio transcriptions. *Pages 307–308 of: Proceedings of the 12th international ACM SIGACCESS conference on Computers and accessibility.* ASSETS '10. New York, NY, USA: ACM.

Whitehill, Jacob, Ruvolo, Paul, Wu, Tingfan, Bergsma, Jacob, and Movellan, Javier. 2009. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. *Pages 2035–2043 of: Advances in Neural Information Processing Systems.*

WHO. 2010. *International Statistical Classification of Diseases and Related Health Problems 10th Revision (ICD-10).* `http://apps.who.int/classifications/icd10/browse/2010/en`, last accessed on February 3, 2013.

WHO. 2013. *International Classification of Diseases (ICD).* `http://www.who.int/classifications/icd/en/`, last accessed on February 3, 2013.

Wichmann, Pascal. 2011. *Cost-effective information quality improvement in multi-labelling scenarios using human-based electronic services*. Diploma thesis, Karlsruhe Institute of Technology.

Wichmann, Pascal, Borek, Alexander, Kern, Robert, Woodall, Philip, Parlikad, Ajith Kumar, and Satzger, Gerhard. 2011. Exploring the ”Crowd” as enabler of better information quality. *In: Proceedings of the 16th International Conference on Information Quality*.

Yang, Yang, Zhu, Bin B, Guo, Rui, Yang, Linjun, Li, Shipeng, and Yu, Nenghai. 2008. A comprehensive human computation framework: with application to image labeling. *Pages 479–488 of: Proceedings of the 16th ACM international conference on Multimedia*. MM ’08. New York, NY, USA: ACM.

Yuen, Man-Ching, King, Irwin, and Leung, Kwong-Sak. 2011. Task matching in crowdsourcing. *Pages 409–412 of: Internet of Things (iThings/CPSCom), 2011 International Conference on and 4th International Conference on Cyber, Physical and Social Computing*. IEEE.

Zhou, Xiao-Hua, Obuchowski, Nancy A., and McClish, Donna K. 2002. *Statistical methods in diagnostic medicine*. 1st edn. New York, NY, USA: John Wiley and Sons.

Appendix

A. Complementary figures

Table A.1.: Sample space of a scenario with 3 workers and 3 possible responses. A historical failure rate of $p = 0.1$ was assumed for all workers.

#	Result set R	Correctness profile C								
		(0,0,0)	(0,0,1)	(0,1,0)	(0,1,1)	(1,0,0)	(1,0,1)	(1,1,0)	(1,1,1)	$P_E(R)$
1	$a_1a_1a_1$	0.00005	0	0	0	0	0	0	0.24300	0.24305
2	$a_1a_1a_2$	0.00005	0.00075	0	0	0	0	0.01350	0	0.01430
3	$a_1a_1a_3$	0.00005	0.00075	0	0	0	0	0.01350	0	0.01430
4	$a_1a_2a_1$	0.00005	0	0.00075	0	0	0.01350	0	0	0.01430
5	$a_1a_2a_2$	0.00005	0	0	0.01350	0.00075	0	0	0	0.01430
6	$a_1a_2a_3$	0	0.00075	0.00075	0	0.00075	0	0	0	0.00225
7	$a_1a_3a_1$	0.00005	0	0.00075	0	0	0.01350	0	0	0.01430
8	$a_1a_3a_2$	0	0.00075	0.00075	0	0.00075	0	0	0	0.00225
9	$a_1a_3a_3$	0.00005	0	0	0.01350	0.00075	0	0	0	0.01430
10	$a_2a_1a_1$	0.00005	0	0	0.01350	0.00075	0	0	0	0.01430
11	$a_2a_1a_2$	0.00005	0	0.00075	0	0	0.01350	0	0	0.01430
12	$a_2a_1a_3$	0	0.00075	0.00075	0	0.00075	0	0	0	0.00225
13	$a_2a_2a_1$	0.00005	0.00075	0	0	0	0	0.01350	0	0.01430
14	$a_2a_2a_2$	0.00005	0	0	0	0	0	0	0.24300	0.24305
15	$a_2a_2a_3$	0.00005	0.00075	0	0	0	0	0.01350	0	0.01430
16	$a_2a_3a_1$	0	0.00075	0.00075	0	0.00075	0	0	0	0.00225
17	$a_2a_3a_2$	0.00005	0	0.00075	0	0	0.01350	0	0	0.01430
18	$a_2a_3a_3$	0.00005	0	0	0.01350	0.00075	0	0	0	0.01430
19	$a_3a_1a_1$	0.00005	0	0	0.01350	0.00075	0	0	0	0.01430
20	$a_3a_1a_2$	0	0.00075	0.00075	0	0.00075	0	0	0	0.00225
21	$a_3a_1a_3$	0.00005	0	0.00075	0	0	0.01350	0	0	0.01430
22	$a_3a_2a_1$	0	0.00075	0.00075	0	0.00075	0	0	0	0.00225
23	$a_3a_2a_2$	0.00005	0	0	0.01350	0.00075	0	0	0	0.01430
24	$a_3a_2a_3$	0.00005	0	0.00075	0	0	0.01350	0	0	0.01430
25	$a_3a_3a_1$	0.00005	0.00075	0	0	0	0	0.01350	0	0.01430
26	$a_3a_3a_2$	0.00005	0.00075	0	0	0	0	0.01350	0	0.01430
27	$a_3a_3a_3$	0.00005	0	0	0	0	0	0	0.24300	0.24305
	$P_E(C)$	0.00100	0.00900	0.00900	0.08100	0.00900	0.08100	0.08100	0.72900	1.00000

Table A.2.: Conditional probability $P(C \mid R)$ for observing a certain correctness profile C given specific response tuple R for the sample space in table A.1.

#	Result set R	Conditional probability $P_E(C \mid R)$ for correctness profile C							
		(0,0,0)	(0,0,1)	(0,1,0)	(0,1,1)	(1,0,0)	(1,0,1)	(1,1,0)	(1,1,1)
1	$a_1a_1a_1$	0.0002	0	0	0	0	0	0	0.9998
2	$a_1a_1a_2$	0.00333	0.05246	0	0	0	0	0.94421	0
3	$a_1a_1a_3$	0.00333	0.05246	0	0	0	0	0.94421	0
4	$a_1a_2a_1$	0.00333	0	0.05246	0	0	0.94421	0	0
5	$a_1a_2a_2$	0.00333	0	0	0.94421	0.05246	0	0	0
6	$a_1a_2a_3$	0	0.33333	0.33333	0	0.33333	0	0	0
7	$a_1a_3a_1$	0.00333	0	0.05246	0	0	0.94421	0	0
8	$a_1a_3a_2$	0	0.33333	0.33333	0	0.33333	0	0	0
9	$a_1a_3a_3$	0.00333	0	0	0.94421	0.05246	0	0	0
10	$a_2a_1a_1$	0.00333	0	0	0.94421	0.05246	0	0	0
11	$a_2a_1a_2$	0.00333	0	0.05246	0	0	0.94421	0	0
12	$a_2a_1a_3$	0	0.33333	0.33333	0	0.33333	0	0	0
13	$a_2a_2a_1$	0.00333	0.05246	0	0	0	0	0.94421	0
14	$a_2a_2a_2$	0.0002	0	0	0	0	0	0	0.9998
15	$a_2a_2a_3$	0.00333	0.05246	0	0	0	0	0.94421	0
16	$a_2a_3a_1$	0	0.33333	0.33333	0	0.33333	0	0	0
17	$a_2a_3a_2$	0.00333	0	0.05246	0	0	0.94421	0	0
18	$a_2a_3a_3$	0.00333	0	0	0.94421	0.05246	0	0	0
19	$a_3a_1a_1$	0.00333	0	0	0.94421	0.05246	0	0	0
20	$a_3a_1a_2$	0	0.33333	0.33333	0	0.33333	0	0	0
21	$a_3a_1a_3$	0.00333	0	0.05246	0	0	0.94421	0	0
22	$a_3a_2a_1$	0	0.33333	0.33333	0	0.33333	0	0	0
23	$a_3a_2a_2$	0.00333	0	0	0.94421	0.05246	0	0	0
24	$a_3a_2a_3$	0.00333	0	0.05246	0	0	0.94421	0	0
25	$a_3a_3a_1$	0.00333	0.05246	0	0	0	0	0.94421	0
26	$a_3a_3a_2$	0.00333	0.05246	0	0	0	0	0.94421	0
27	$a_3a_3a_3$	0.0002	0	0	0	0	0	0	0.9998

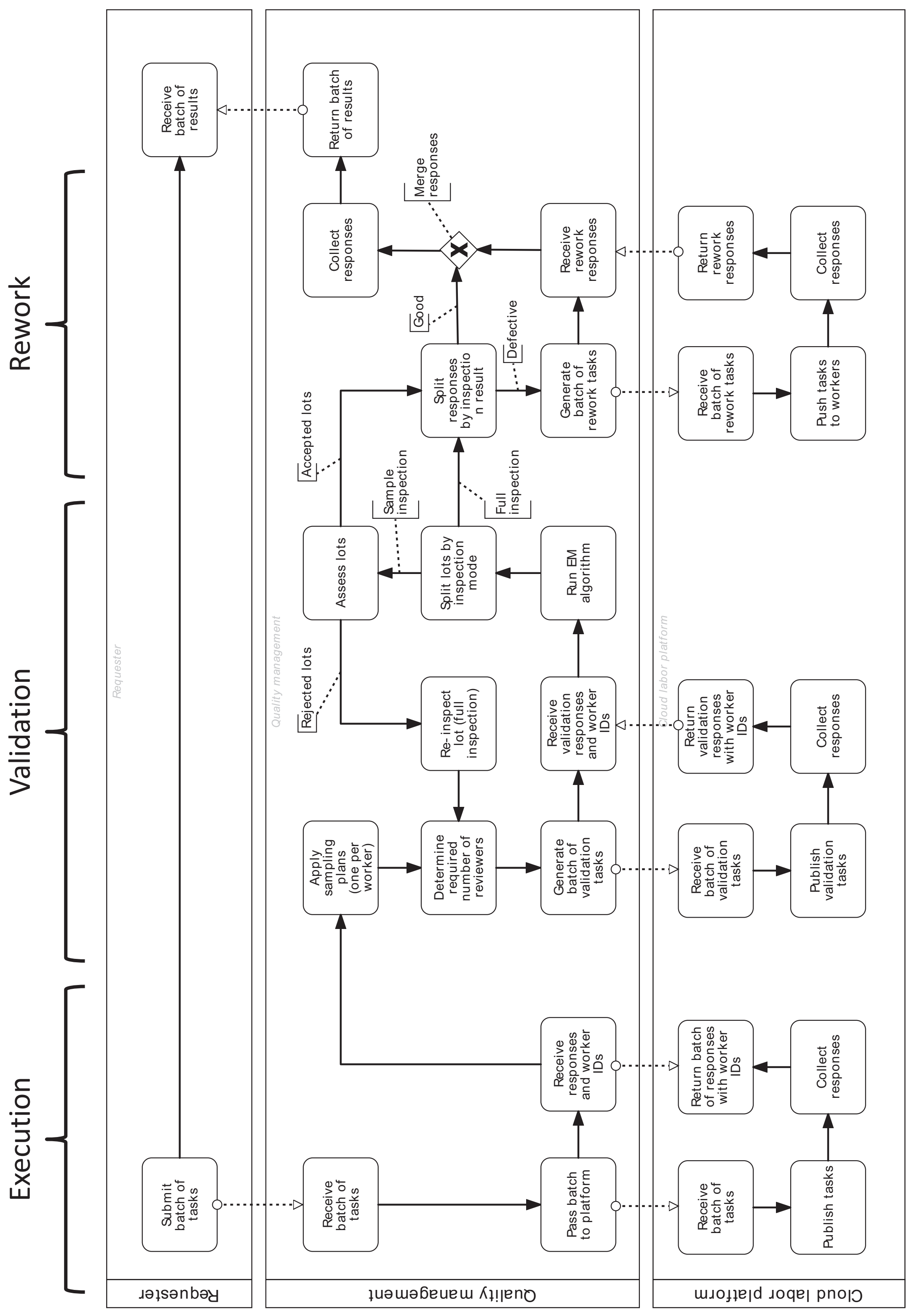
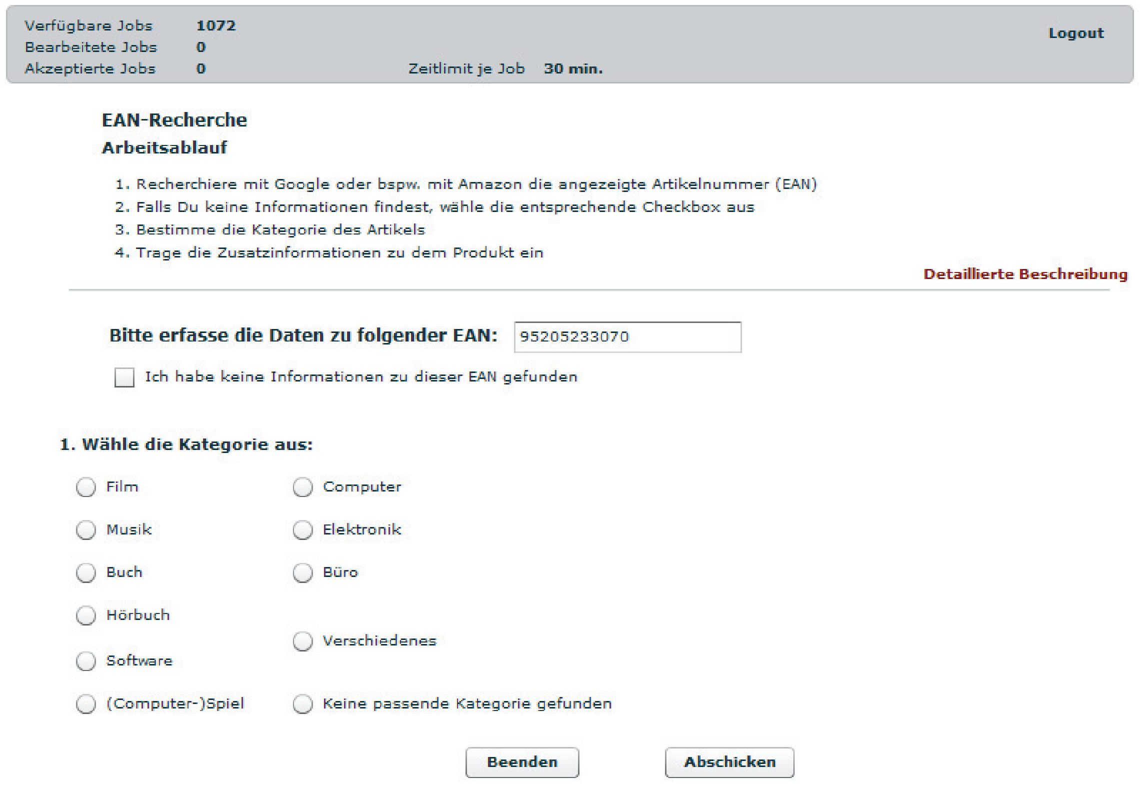


Figure A.1.: Process flow of the group validation approach.



Verfügbare Jobs 1072
Bearbeitete Jobs 0
Akzeptierte Jobs 0
Zeitlimit je Job 30 min.

EAN-Recherche
Arbeitsablauf

1. Recherchiere mit Google oder bspw. mit Amazon die angezeigte Artikelnummer (EAN)
2. Falls Du keine Informationen findest, wähle die entsprechende Checkbox aus
3. Bestimme die Kategorie des Artikels
4. Trage die Zusatzinformationen zu dem Produkt ein

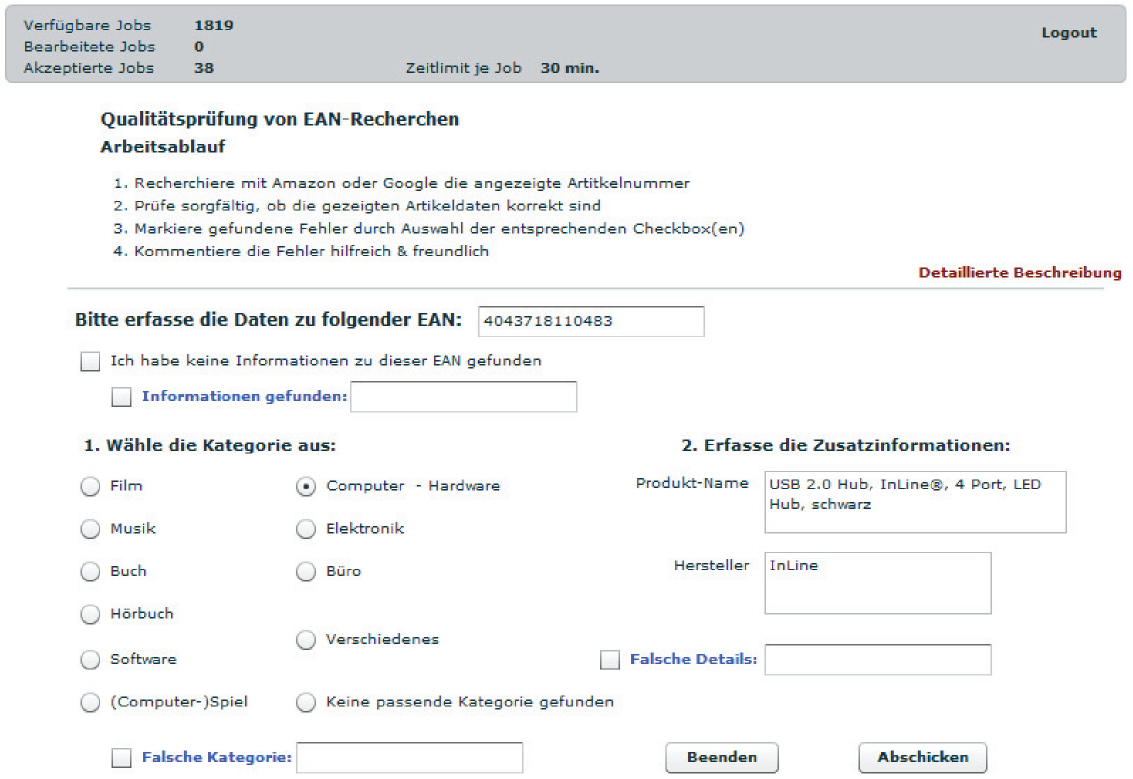
Bitte erfasse die Daten zu folgender EAN: 95205233070

☐ Ich habe keine Informationen zu dieser EAN gefunden

1. Wähle die Kategorie aus:

○ Film ○ Computer
○ Musik ○ Elektronik
○ Buch ○ Büro
○ Hörbuch ○ Verschiedenes
○ Software
○ (Computer-)Spiel ○ Keine passende Kategorie gefunden

Figure A.2.: Screenshot of the product research scenario data acquisition task interface.



Verfügbare Jobs 1819
Bearbeitete Jobs 0
Akzeptierte Jobs 38
Zeitlimit je Job 30 min.

Qualitätsprüfung von EAN-Recherchen

Arbeitsablauf

1. Recherchiere mit Amazon oder Google die angezeigte Artikelnummer
2. Prüfe sorgfältig, ob die gezeigten Artikeldaten korrekt sind
3. Markiere gefundene Fehler durch Auswahl der entsprechenden Checkbox(en)
4. Kommentiere die Fehler hilfreich & freundlich

Bitte erfasse die Daten zu folgender EAN: 4043718110483

☐ Ich habe keine Informationen zu dieser EAN gefunden

☐ **Informationen gefunden:**

1. Wähle die Kategorie aus:

○ Film
○ Musik
○ Buch
○ Hörbuch
○ Software
○ (Computer-)Spiel
[✓ Computer - Hardware]
○ Elektronik
○ Büro
○ Verschiedenes
○ Keine passende Kategorie gefunden

☐ **Falsche Kategorie:**

2. Erfasse die Zusatzinformationen:

Produkt-Name: USB 2.0 Hub, InLine®, 4 Port, LED Hub, schwarz

Hersteller: InLine

☐ **Falsche Details:**

Figure A.3.: Screenshot of the product research scenario validation task interface.

B. Complementary research

As an alternative to the goal-based QM for cloud labor services presented in the thesis, this appendix provides initial considerations for applying cost-benefit objectives which have been motivated in section 10. As a starting point, the multi-labeling scenario introduced in section 6.1 is being used.

The approach presented here uses the concept of *value of information* (Marschak & Radner, 1958) as the decision-theoretical foundation to decide whether in the process of the DVM for multi-labeling, further redundancy is required or not. Instead of a well-defined correctness goal, the requester specifies a cost-benefit objective which is basically a compromise between cost and quality.

Utility function

The cost-benefit objective is represented by a utility function that addresses the potential benefit of getting a label right combined with the potential costs of mislabeling. It is defined as a basic 2×2 matrix that assigns a constant value to each combination of true label $t_x \in \{0, 1\}$ and overall labelling decision $r_x \in \{0, 1\}$:

$$u(t_x, r_x) = \begin{pmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{pmatrix} = \begin{pmatrix} \textit{utility of TN} & \textit{utility of FP} \\ \textit{utility of FN} & \textit{utility of TP} \end{pmatrix} \tag{10.1}$$

The utility matrix consists of four values that capture the utilities of TPs, TNs, FPs and FNs. The meaning of the first two utilities is obvious: They represent the economic benefit that the requesters gain when obtaining correct information about a label t_x. The latter two values represent the typically lower or even negative utilities of type I and type II errors (FPs and FNs). The difference between the obtained utility of an FP and the utility of the TN that could have been obtained can be seen as a penalty for the FP. Same applies to the FNs versus TPs. Even in cases in which a labeling error does not result in direct costs, the opportunity costs caused by the lost profit due to an incorrectly identified label are considered. Both direct as well as indirect costs of incorrect labeling should be regarded.

Labeling decision

As defined by (6.4), for a specific label x the probability that it has the true value LP_x given the observed evidence is $P(t_x = 1 | r_{x1}, .., r_{xw})$. The decision threshold, the so-called *equilibrium probability*(EP), is the posterior probability for which assigning or non-assigning a label results in the same expected utility U_x:

$$EP = LP_x \Leftrightarrow U_x(r_x = 0) = U_x(r_x = 1) \tag{10.2}$$

with

$$U_x(r_x) = (1 - LP_x) \cdot u(t_x = 0, r_x) + LP_x \cdot u(t_x = 1, r_x). \tag{10.3}$$

With (10.1), this leads to:

$$EP = \frac{u_{00} - u_{01}}{u_{00} + u_{11} - u_{10} - u_{01}} \tag{10.4}$$

EP solely depends on the values in the utility matrix. As the utility values for the TPs and TNs will be higher than the ones for the FPs and FNs, the denominator will not be zero.

The above equation is the minimum probability LP_x that must have been reached before a label is assigned in order to obtain the maximum expected utility. Therefore, the consolidated labeling decision for label x is:

$$r_x = \begin{cases} 0 \text{ if } LP_x < EP \\ 1 \text{ if } LP_x \geq EP \end{cases} \tag{10.5}$$

The probability EP represents the discrimination threshold that translates the continuous posterior probability for each label into a binary decision on whether to accept or reject a label.

Value-of-information

The idea of the VoI concept is to decide whether the cost for acquiring additional information justifies the value of the information provided by an additional worker.

The evidence that has been observed up to the point of decision is captured by the decisions of the w workers represented by the matrix R in (10.1). For a specific label x, the evidence is LP_x. It can be seen as the subjective belief we have with respect to a label prior to observing the labeling decision of an additional worker. The labeling decision this additional worker may make is denoted by the vector $G_v = (g_1, .., g_v) \in \{0,1\}^v$. For each label x, the value $g_x \in \{0,1\}$ represents the worker's (yet unknown) labeling decision.

The value V of observing additional information (before considering cost of information) is defined as the difference between the expected utility with that information $U(R,G)$ and the expected utility without that information $U(R)$. For the labeling scenario that is:

$$V_R(G) = U(R,G) - U(R) \tag{10.6}$$

Because of the independence of the labels, the overall utilities are calculated as the sum over the utilities for each label x:

$$V_R(G) = \sum_{x=1}^{v} V_{r_{x1},..,r_{xw}}(g_x) = \sum_{x=1}^{v} (U_x(r_{x1}, .., r_{xw}, g_x) - U_x(r_{x1}, .., r_{xw})) \tag{10.7}$$

The expected utility without additional information $U_x(r_x)$ is defined by (10.3) and (10.5):

$$U_x(r_{x1}, .., r_{xw}) = \begin{cases} (1 - LP_x) \cdot u(t_x = 0, r_x = 0) + LP_x \cdot u(t_x = 1, r_x = 0) \text{ if } LP_x < EP \\ (1 - LP_x) \cdot u(t_x = 0, r_x = 1) + LP_x \cdot u(t_x = 1, r_x = 1) \text{ if } LP_x \geq EP \end{cases} \tag{10.8}$$

In order to calculate the expected utility $U_x(r_{x1}, .., r_{xw}, g_x)$ with additional information, estimates of the labeling performance parameters $\alpha_(w + 1)$ and $\beta_(w + 1)$ of the additional worker $w + 1$ are required. If the worker performing the next assignment is not known in advance, these parameters have to be estimated. Because the value that g_x will take is uncertain, both cases have to be considered with respect to the overall decision for label x denoted by $r_x \in \{0,1\}$:

$$\begin{aligned} U_x(r_{x1}, .., r_{xw}, g_x) =& P(g_x = 0) \cdot U_x(r_{x1}, .., r_{xw}, r_{x,w+1} = 0) + \\ & P(g_x = 1) \cdot U_x(r_{x1}, .., r_{xw}, r_{x,w+1} = 1) \end{aligned} \tag{10.9}$$

The value of $U_x(r_{x1}, .., r_{xw}, r_{x,w+1})$ is calculated with (10.8) assuming that there are $w+1$ workers. The marginal likelihood $P(g_x)$ is computed as:

$$\begin{aligned} P(g_x) &= P(g_x = 0|t_x = 0) \cdot P(t_x = 0) + P(g_x = 1|t_x = 1) \cdot P(t_x = 1) \\ &= \alpha_(w + 1) \cdot (1 - \tilde{p}) + \beta_(w + 1) \cdot \tilde{p} \end{aligned} \tag{10.10}$$

with $\alpha_(w{+}1)$ and $\beta_(w{+}1)$ being the (estimated) sensitivity and specificity of the additional worker. The probabilities $\tilde{p}$ and $((1 - \tilde{p})$ are the prior probabilities for a label to be true or false according to (6.10).

Decision on additional redundancy

In order to decide whether feedback should be requested from another worker, the expected value $V_R(G)$ of that information needs to be compared to the costs $C(G)$ associated with an additional labeling task. If the expected value exceeds the costs, i.e. $V_R(G) > C(G)$, redundancy is being increased, otherwise the supposedly true labels identified so far are being returned to the requester. The value $V_R(G)$ is being calculated according to (10.6), the costs $C(G)$ are determined depending on the scenario and the setup of the human computation task.

Results

In order to validate the WMV-ML mechanism with a cost-benefit objective, a fraud detection scenario was assumed in which costs are associated to missing codes (FNs) or incorrect codes (FPs). The average costs for an FP was assumed to be 0.70 EUR based on considerations about the associated average risk of fraud and the expected loss. The cost for an FN was assumed to be 0.40 EUR covering the additional effort for manually revising the suspicion. The resulting utility matrix is:

$$u(t_x, r_x) = \begin{pmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{pmatrix} = \begin{pmatrix} 0.00 & -0.70 \\ -0.40 & 0.00 \end{pmatrix} \tag{10.11}$$

For the labeling effort, a cost of 0.055 EUR was assumed per ICD-10 code. As the tasks are allocated to the workers in form of an open call, the performance of the projected additional worker is not known in advance. Hence, it has been estimated by the average performance of all available workers that have not submitted a result for this task yet. The averaged results of 10 experiments are presented in the last row of Table 3. The average redundancy per HIT was 2.3. On average 2.7 out of 45 tasks were not completed due to an insufficient number of results. An average sensitivity of 0.912 and an average specificity of 0.942 were achieved. Compared to the evaluation of the DVM for multi-labeling presented in section 9.1, the sensitivity and specificity are even higher at a slightly lower degree of redundancy i.e. at lower cost.

Discussion

Although, a cost-benefit scenario may appear to be a great approach for the medical coding scenario, it turned out that it did not provide a reasonable solution in practice. The reasons are twofold: First, the ICD-10 codes resulting from the coding process are considered to be a general asset for the health insurance provider that can serve various use cases. Not all future use cases may be obvious at present and a consistent coding quality over a period of several years may be needed as a basis for them. However, the

Table B.1.: Results of the DVM for multi-labeling with a cost-benefit objective being applied to a medical coding scenario.

Experiment number	Average Sensitivity	Average Specificity	Number of Results	Average Redundancy	Unfinished
1	0.869	0.956	103	2.289	3
2	0.923	0.953	104	2.311	2
3	0.919	0.905	105	2.333	4
4	0.895	0.947	102	2.267	3
5	0.927	0.944	109	2.422	2
6	0.925	0.956	108	2.4	3
7	0.921	0.943	102	2.267	3
8	0.913	0.952	104	2.311	3
9	0.932	0.919	101	2.244	2
10	0.893	0.94	99	2.2	2
Average	0.912	0.942	103.7	2.304	2.7

utilities of the ICD-10 codes heavily depend on the use case. Therefore, it is difficult to define all-purpose utilities that also foresee future scenarios.

Another challenge is that a non-traditional mindset is required in order to agree on a cost-benefit objective in a service-consumer service-provider scenario because the benefit of the service is rather perceived by the consumer while its costs are mainly perceived by the provider. To come to an agreement, both the provider and the consumer would likely need to disclose their cost structure. The challenge of defining cost-optimal SLAs has recently being addressed by several research papers, e.g. by (Kieninger et al., 2012). However, for cloud labor services it appeared to be a more fundamental need to address well-defined quality goals first.