The final authenticated version is available online at
https://doi.org/10.1007/978-3-030-00761-4_3

Guidance of Architectural Changes in Technical Systems with Varying Operational Modes

Lukas Märtin[1], Nils-André Forjahn[1],
Anne Koziolek[2], and Ralf Reussner[2]

[1] TU Braunschweig, Braunschweig, Germany,
{l.maertin,n.forjahn}@tu-braunschweig.de
[2] Karlsruhe Institute of Technology, Karlsruhe, Germany,
{koziolek,reussner}@kit.edu

Abstract. Technical systems often rely on redundant platforms. One way to increase dependability is to define various QoS modes, applied to different hardware resources. Switching between modes is limited by resource availability and causes costs for structural changes. Hence, selecting appropriate system architectures for specific resource sets and defining cost-efficient mode sequences is challenging. This short paper proposes an approach to support reconfiguration decisions for varying modes. We extend our decision graphs for traversing architectures towards multi-purpose applicability. We optimise reconfigurations within individual modes while reducing costs of mode changes simultaneously. Graph-based differentiations lead to most efficient mode sequences, transition configurations and visualisations. To respect high reconfigurability, we particularly inspect impacts of resource faults. For evaluation, we apply a subsystem of a micro satellite with multiple operational modes.

1 Introduction

Developing *technical systems* often incorporates grouping of redundant resources. Each feasible *system configuration* is evaluated by quality demands wrt. an *operational mode*. A *multi-purpose system* executes tasks in varying modes and redundancy groups. Due to modes changes and resources faults this is challenging. Thus, for cost-efficient maintenance, it is essential to identify relations between modes and optimal transitions. Here, we propose to prioritise configurations according architectural quality at design time to reduce maintenance costs. Thus, a prioritised reconfiguration space is tailored to modes and continuously synchronised to resource availability for reducing efforts of manual exploration.

In previous work, we introduced an architecture-oriented approach to support reconfiguration decisions, the *Deterministic Architecture Relation Graph* (*DARG*) [10]. Our existing approach is integrated [9] with the concept of *Degrees of Freedom* [6] for architecture optimisation to generate quality-accessed configurations from large decision spaces.

In this short paper, we refine our work towards multi-purpose application. Thus, an *DARG* instance is exchanged for each mode change and appropriate

transition configurations are explored. To define a cost-efficient *ordering of modes* and *efficient transitions*, we inspect commonalities and efficient reconfigurations in intersections of *DARG* pairs. For evaluation, we apply our approach to a satellite subsystem with varying resource constellations and multiple modes. For dependability, the result stability is checked to assure a fault-aware solution.

The remainder of this paper is organised as follows. Sect. 2 summarises related work. Our approach is described in Sect. 3 and evaluated in Sect. 4. We conclude and promote future work in Sect. 5.

2 Related Work

We make use of knowledge from configuration generator to relate alternate solutions with slight differences in resource demands and qualities. Similar to that, Barnes et al. [1] define relations between architectures on candidate evolution paths. These paths specify a search-based reconfiguration process from a source to a user-defined target architecture via a sequence of transient architectures. We intend to explore such targets automatically. Jung et al. [5] determine policies to adapt running systems in the cloud. A decision-tree learner that is trained with feasible system configurations, generated by queuing models, derives these policies at design time. Close to that, Frey et al. [4] inspect reconfigurations as deployment options derived by genetic algorithms. The authors define rules at design time to systematically change the deployment of a system at run time. Both approaches neither explicitly represent the reconfiguration space nor the qualitative trade-offs between design options. Our approach preserves knowledge from the configuration generation to prioritise near-optimal candidates. Malek et al. [8] proposes a more hardware-oriented approach in context of self-adaptive systems. The authors provide a trade-off model to identify a suitable deployment architecture. Even though that approach is applicable to distinguish between configurations, the authors did not integrate a prioritisation of all feasible design alternatives for decision support at run time.

3 Reconfiguration Support for Varying Operation Modes

We extend our work by multi-purpose capabilities to reduce maintenance efforts.

3.1 Baseline Models

Due to space constraints all models [9, 10] are summarised here. A resource platform *RP* defines properties and constraints of resources. Redundancies are represented by groups. Properties provide value assignments for quality attributes from the configuration generation. We model configurations as sets of software components with bindings *bind* to resources in *RP*. A configuration has a unique *id* and holds aggregated values *val* for attributes. An *ARG* defines an unsorted reconfiguration space. Its nodes are parametrised by generated configurations

and their bindings to RP. Edges relate configurations via transient nodes or directly. Changes in resource bindings and reconfiguration costs are annotated as edge labels. Initially an ARG is ambiguous as labels might be non-deterministic. To fix that, we adapt *Quality Attribute Directed Acyclic Graphs* [3] ($QADAG$) to describe operational modes as weighted sums with constraints for attributes. Values for attributes are extracted from ARG nodes. For aggregation, each value is normalised. The user customises a $QADAG$ wrt. an operational mode by setting weightings and *minimal acceptance values*. A value drops to 0 if its minimal acceptance is violated. $DARG$ instances are derived as mode-specific refinements of the ARG for each $QADAG$ instance by evaluating a *utility* for all configuration results. Hence, edges are qualitatively definable now. A $DARG$ instance is an architecture-based model of the *reconfiguration space* for a specific mode.

3.2 Aligning Changes of Operational Modes and DARG Instances

The interrelationships of multiple $QADAG$ and $DARG$ instances are considered by deriving optimal orders. Built on existing analysis methods a $QADAG$ is customised to derive a corresponding $DARG$ instance. The *multi-mode analysis* inspects the importance of a configuration within its origin instance and extracts an optimal order between all instances. Faults are injected to check robustness.

Prioritise Configurations for Mode Transitions. Each configuration in a $DARG$ instance is rated by its fitness for mode transitions. In addition to existing indicators, we apply *centralities* to specify the intra-graph importance of each node. According to graph size, the inexpensive *degree centrality*, the path-oriented *closeness centrality* or the expensive graph-spanning *betweenness centrality* might be applied.

Three *transition criteria* result to rate reconfiguration abilities. The (1) normalised *utility* defines the fitness to the mode, the (2) amount of resource bindings *bind* characterises the degree of redundancies and the (3) centrality *cen* of the corresponding node quantises the reconfiguration options wrt. related configurations. The resource amounts and centrality is normalised over all occurrences. By setting constant weighting the criteria are prioritised. Each configuration c_i is rated by a *transition value* in its reconfiguration space D_i, defined by $tv_{c_i,D_i} := w_1 * utility_{c_i} + w_2 * cen_{c_i} + w_3 * |bind_{c_i}|$.

Extracting Optimal Mode Sequences. For extracting optimal mode sequences, we inspect all transition values of configurations in pairs of $DARG$ instances. Because of the common ARG, the configurations of $DARG$ instances intersect, matched by *id*. Let D_s and D_t be reconfiguration spaces in $\mathcal{D}$, then $D_s \cap_{id} D_t \iff \forall c_i \in D_s, c_j \in D_t : {c_i}_{id} == {c_j}_{id}$ is the configuration intersection of the pair. Although it is unlikely that an intersection is empty, then an ordering of this pair is infeasible and is done randomly instead. To inspect the transition values between reconfiguration spaces, we aggregate the values to sums. Due to asymmetries in the values wrt. the source spaces, we inspect two values per pair.

For the pair of reconfiguration spaces D_s and D_t the *transition value sum* is defined by $tvs_{D_s \cap_{id} D_t} := \left(\sum_{i=1}^{|D_s \cap_{id} D_t|} tv_{c_i,D_s} + tv_{c_i,D_t} \right) \mid c_i \in D_s \cap_{id} D_t$.

To explore all pairs, we permute over D by $S \subset \mathcal{P}(\mathcal{D})$ with $S_i \in S$ is an *ordered sequence* and $|S_i| == |\mathcal{D}|$ the permutation size equals the amount of all $DARG$ instances. All transition value sums in S are inspected for the overall maximum by $tvs_{max} := \max \sum_{i=1}^{|S|} (tvs_{D_s \cap_{id} D_t} \forall D_s, D_t \in S_i) \mid S_i \in S$.

The highest sum characterises the *optimal sequence* of all $DARG$ instances with the set of *transition configurations*. By backtracking $DARG$ to $QADAG$ instances, the order of reconfiguration spaces lead to the *optimal mode sequence*.

Consideration of Resource Faults. Each resource fault reduces the amount of configurations. If all options in a redundancy group are affected, a configuration is invalid. This causes changes in commonalities between reconfiguration spaces as well as centralities within the graphs. Thus, faults have significant impacts on the selection of transition configurations and optimal mode sequences. Hence, expected resource faults are injected in $DARG$ instances. The transition configurations and the optimal sequence are checked for stability in an iterative process. If a pair of $DARG$ instances is affected by a fault in the resource platform, it is marked for re-ordering and all transition values are re-calculated. The ordering might be updated due to the new maximum transition value sums.

4 Validation

We validate our approach along a subsystem of the TET-1 satellite.

4.1 Application Scenario

TET-1 is designed for verifying experimental hardware. We apply our approach to the *attitude control system* (ACS) [7]. The ACS architecture is threefold. Sensor resources *estimate* the position and orientation then necessary attitude changes are *predicted* and at least required actions are *controlled* on actuation resources. Such resource dependencies are represented by components on the lowest level. To enhance reconfiguration abilities for heterogeneous redundancies, we relaxed constraints and added sensing variations to the original design. For actuation two groups with reaction wheels (RWS) and magnetic coils (MCS) exist. Sensing is performed by five groups consisting of star compasses (ASC), sun sensors (CSS), magnetic field sensors (MFS), inertial measurement units (IMU) and on-board navigation systems (ONS). We model the ACS as variant-rich *Palladio Component Model* [2] instance with several degrees of freedom with Palladio DSE. Our tool AREva[3] generates an ARG model and a default $QADAG$. We apply three experiments [7] with varying modes: A Li-Polymer battery ($N1$), a pico propulsion system ($N7$) and an infra-red camera ($N15$). Based on six quality attributes and data sheets, three $QADAG$ with corresponding $DARG$ instances are derived for validation. For sake of space, we leave out details.

[3] AREva tool and **validation data**, `https://www.github.com/lmaertin/areva`

4.2 Design of Experiment

The exploration is challenging if operational modes vary and faults occur. Thus, we (1) define optimal mode sequences and (2) approve its fault robustness.

The importance of a configuration in a *DARG* is vital to define appropriate transition configurations for sequencing modes. Maximum classification values for configurations are identified to differentiate pairs of reconfiguration spaces efficiently. For that, we aggregate transition values within each *DARG* instance to rate orders of mode sequences. The maximum sums over all transition values in intersections of all pairs of reconfiguration spaces are calculated. By ordering of the sums, an optimal mode sequence is determined.

High reconfigurability is only assured if the mode sequence is robust against faults. Thus, if a fault affects a valid configuration the corresponding mode sequence must hold. The impact of faults on the validity of mode ordering is examined on level of relative changes of transition values. The ordering needs to be proven as stable. Here, we adapt the previous fault-less measurements to estimate impacts of faults. For that, we inspect the distance of changes in the sums of transition values. The order of these relative values is checked for compliance to the recent order of modes. If the maximum changes, the mode sequence is no longer stable and a re-ordering is initiated.

4.3 Measurements and Explanation

Our measurement are done on basis of our ACS model and the sequence of expected faults from Tab. 1. For each experiment, we choose an initial configuration with a high amount of resources to enhance reconfigurability.

1) Identifying Optimal Mode Sequences. We perform a multi-mode analysis on basis of *DARG* instances for each experiment. At first, we explore transition configurations and optimal mode sequences in a faultless setting. Afterwards, we inject a fault sequence and observe impacts on the initial results. For both settings, the calculation of the transition values is parametrised to 0.33 for w_1, w_2 and w_3. We applied all three kinds of centralities. Even though all measurements perform well, we show the results for betweenness due to lack of space. The analysis calculates individual transition values and explores maximum sums in *six possible orders* of intersecting *DARG* instances. For each order, the transition value sum consists of sums from two transitions. The order $N15 \rightarrow N1 \rightarrow N7$ has the highest sum and is the optimal mode sequence when no faults occur. It is followed by $N1 \rightarrow N15 \rightarrow N7$ and $N7 \rightarrow N1 \rightarrow N15$.

For explanation, we take a deeper look into the amount of transition configurations between a pair of modes and their transition value distribution. A transition between $N1$ and $N7$, and vice versa, has the most transition configurations with 86. $N17 \leftrightarrow N15$ has a sum of 85 and $N77 \rightarrow N15$ at least 77. Therefore, mode orders containing transitions between $N1$ and $N7$ are more likely to have higher transition value sums than other transitions if the transition value distributions are similar. This means that reconfigurations due to mode transitions

between $N1$ and $N7$ are more effective because the reconfiguration spaces are more structurally similar than all other combinations. Fig. 1 shows the transition value distribution of the configurations for each mode transition. While the

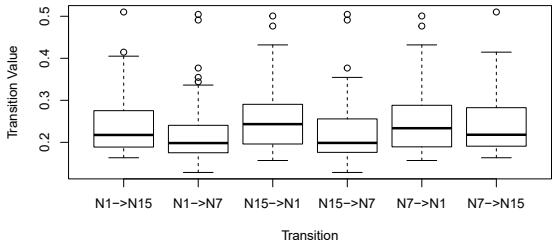

Fig. 1. Transition Value Distribution per Transition Direction

medians and value ranges for each transition are different, they are still within similar value ranges. For instance, the transition $N15 \rightarrow N1$ provides higher rated transition configurations in average. Therefore, this transition probably leads to highly rated configurations to perform the experiment $N1$ well. The high amount of transition configurations in both transitions of $N15 \rightarrow N1 \rightarrow N7$ leads to reduced reconfiguration costs. Additionally, high average transition values promote appropriate configurations.

2) Inspecting Fault Robustness. We derived a sequence of faults that effectively harms the ACS wrt. fault analysis methods in Tab. 1. To examine the robustness of our results, the changes of transition value sums for each possible order is shown after fault injection. The initial optimal mode order of

Table 1. Fault Sequence and Impacts on Transition Value Sums

Fault	1→7→15	7→1→15	7→15→1	1→15→7	15→1→7	15→7→1
GPS LNA1+Antenna1	37.33	39.93	37.36	40.03	**41.30**	37.48
GPS Receiver1	37.24	38.65	36.98	40.26	**42.33**	37.03
GPS LNA2+Antenna2	10.09	12.31	11.55	12.86	**13.02**	10.08
MFS Fluxgate1	10.11	12.20	11.61	12.83	**13.07**	10.12
CSS RearHead2	9.90	12.19	11.61	12.78	**12.90**	10.12
ASC DPU1 High Res.	10.15	12.41	11.63	13.01	**13.11**	10.13
ASC DPU2 Low Res.	10.07	12.21	11.53	12.77	**13.04**	10.03
CSS Chipset1 High Res.	9.88	12.07	11.49	12.65	**12.78**	10.06
CSS RearHead1	2.15	3.28	2.92	3.11	**3.36**	2.10
CSS FrontHead2	2.14	3.27	2.93	3.10	**3.35**	2.10
CSS Chipset2 Low Res.	2.13	3.27	2.92	3.11	**3.34**	2.10
RW1	1.84	2.68	2.60	**2.78**	2.74	1.81
MC1x	0.58	0.74	0.73	**0.76**	0.76	0.57
RW2, MC2y, MC2x	-, -, -	-, -, -	-, -, -	-, -, -	-, -, -	-, -, -

$N15 \rightarrow N1 \rightarrow N7$ that was calculated under a faultless state is stable for most of the injected faults. However, when RW1 fails the transition value sum of

$N1 \rightarrow N15 \rightarrow N7$ beats the original optimal order. After injecting the fault of RW2, no transition configurations are left and the reconfiguration process stops. Overall, the complete failure of the ONS, CSS and the majority of actuation resources have the biggest impacts on available transition configurations. At the beginning in particular, the redundancy groups are thinned out. If a group becomes disadvantageous or invalid, similar resources from other groups are used. However, other resources in the affected resource groups already failed before, like the first GPS Antenna1+LNA1 or the second CSS RearHead2. That is why the fault of the second GPS LNA2+Antenna2, the CSS RearHead1 and the MC1x led to the complete failure of their corresponding group.

Following, we inspect the reasons for the change of the optimal order after RW1. While the amount of transition configurations has already been heavily reduced by faults before, transitions of $N1 \leftrightarrow N7$ and $N1 \leftrightarrow N15$ are still more beneficial from a structural standpoint with 8 and 13 transition configurations respectively. After the failure of RW1 the transitions of $N1 \leftrightarrow N15$ and $N7 \leftrightarrow N15$ are both equally disadvantageous in terms of structural similarity compared to $N1 \leftrightarrow N15$ with only 6 transition configurations left. Even though, $N15 \rightarrow N1 \rightarrow N7$ overall had more transition configurations than $N1 \rightarrow N15 \rightarrow N7$, now both have the same amount. Therefore, the choice of optimal mode order is completely reliant on the transition value distribution and the highest average. Fig. 2 shows the distribution of transition values per mode transition direction. The plot

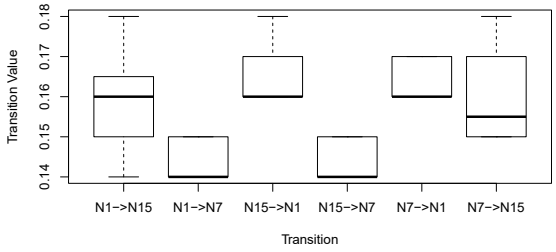

Fig. 2. Transition Value Distribution per Transition Direction

shows that the two partial transitions of $N15 \rightarrow N1 \rightarrow N7$ as well as $N1 \rightarrow N15 \rightarrow N7$ both are very similar. The first transitions $N15 \rightarrow N1$ and $N1 \rightarrow N15$ have almost the same high median. The second transitions $N15 \rightarrow N7$ and $N1 \rightarrow N7$ both have a similar low median. So both orders are also very similar in terms of average transition value. The biggest impact on the final result of ranking $N1 \rightarrow N15 \rightarrow N7$ higher than $N15 \rightarrow N1 \rightarrow N7$ is the first transition value sum, which is sufficient at this point to induce a higher overall sum.

Overall $N15 \rightarrow N1 \rightarrow N7$ should remain as optimal mode sequence because it is stable until the 13^{th} fault. Additionally, it provides the best trade-off between the amounts of similarities between all modes, and thus keeping the reconfiguration cost low, and the average transition values for each configuration, improving the experiments by providing alternate configurations with high utilities.

5 Conclusions

We presented an approach to support maintenance of fault-tolerant technical systems in multi-purpose setting. By performing pre-calculations at design time, we generate knowledge for efficient reconfigurations at run time for varying operational modes. The extended decision model *DARG* supports mode transitions and guides architectural changes by mode sequences. We evaluated our tool-supported approach on a satellite subsystem and proven fault robustness of results. Because of extensive efforts in processing the reconfiguration space, we settled our approach at design time and build upon static data for quality predictions. Consequently, the results might have a lack of precision.

In on-going research, we are doing an empirical study with experts from space industry to explore possible integrations of the approach in the development process. Further improvements are possible by integrating runtime data to continuously update the *DARG*. Here, the computational efforts and delays for reconfiguration at runtime need to be respected to justify our analyses against traditional explorations.

Acknowledgments. This work was partially supported by the DFG under Priority Programme SPP1593: Design For Future – Managed Software Evolution.

References

1. Barnes, J.M., Pandey, A., Garlan, D.: Automated planning for software architecture evolution. In: 28th Int. Conf. on Automated Software Eng. pp. 213–223 (2013)
2. Becker, S., Koziolek, H., Reussner, R.: The palladio component model for model-driven performance prediction. Systems and Software 82(1), 3–22 (2009)
3. Florentz, B., Huhn, M.: Embedded systems architecture: Evaluation and analysis. In: Quality of Softw. Architectures, LNCS, vol. 4214, pp. 145–162. Springer (2006)
4. Frey, S., Fittkau, F., Hasselbring, W.: Search-based genetic optimization for deployment and reconfiguration of software in the cloud. In: 35th Int. Conf. on Software Engineering. pp. 512–521 (2013)
5. Jung, G., Joshi, K., Hiltunen, M., Schlichting, R., Pu, C.: Generating adaptation policies for multi-tier applications in consolidated server environments. In: 5th Int. Conf. on Autonomic Computing. pp. 23–32 (2008)
6. Koziolek, A., Reussner, R.: Towards a generic quality optimisation framework for component-based system models. In: 14th Int. ACM Sigsoft Sympos. on Component based Software Engineering. pp. 103–108 (2011)
7. Löw, S., Herman, J., Schulze, D., Raschke, C.: Modes and more – finding the right attitude for TET-1. In: 12th Int. Conf. on Space Operations (2012)
8. Malek, S., Medvidovic, N., Mikic-Rakic, M.: An extensible framework for improving a distributed software system's deployment architecture. IEEE Trans. on Software Engineering 38(1), 73–100 (2012)
9. Märtin, L., Koziolek, A., Reussner, R.H.: Quality-oriented decision support for maintaining architectures of fault-tolerant space systems. In: 2015 European Conference on Software Architecture Workshops. pp. 49:1–49:5 (2015)
10. Märtin, L., Nicolai, A.: Towards self-reconfiguration of space systems on architectural level based on qualitative ratings. In: 35th Int. Aerospace Conf. (2014)