MODEL DRIVEN DESIGN FOR STRUCTURAL RECONFIGURATION BASED DEPENDABILITY

Imre KOCSIS
Advisor: András PATARICZA

I. Introduction

Large, distributed IT infrastructures providing business-critical services have to protect themselves against internal and external threats and adapt to changing environmental parameters, as load. Most widely applied, structural resilience mechanisms are structure-preserving in the sense that the structure of the system does not change as a reaction. However, recently both the growth in size of distributed systems and technological advancements in various fields as networking technologies and virtualization have been acting as enablers for on-line structural reconfiguration.

This paper introduces the generic concept of structural reconfiguration, explores its modeling and design aspects and outlines a model based, integrated configuration and reconfiguration design workflow for virtualized data centers. Following that, verification aspects of reconfiguration based IT resilience are reviewed and some initial results are described.

II. Structural Reconfiguration Based Resilience of IT Systems

Structural reconfiguration is reallocation of computation, storage or communication that is not only a local, parametric reconfiguration of a system component, but one that changes the service delivery structure and the dependencies between the components in runtime. Industrial practice employs numerous standard mechanisms that are structural reconfiguration approaches, as for example server node failover to hot/cold spares in case of physical or software failure, dynamic reallocation of tasks accompanied with software redistribution in data centers upon changing load and rerouting of communication paths based on changing load or connection failures.

While these and other approaches are common practice, research for the systematic handling of structural reconfiguration based resilience of IT systems has begun only lately with the widespread adoption of the key enabling technologies.

A. Structural Configuration and Structural Reconfigurations

Capturing IT system component and system configuration is a domain-specific modeling task: configuration concepts vary largely among IT management domains. Although standards and quasi-standards exist for configuration modeling, most notably the management data metamodels of system management standards and the vendor-specific metamodels of configuration management databases (see e.g. [1] and [2]), even their typical usage is to utilize only a restricted metamodel of interest with custom extensions.

The *structural configuration* of an IT system is the set of relationships between its components as data flows, resource usage or data carrier relationships, server/client relationships, and so on. Meanwhile, individual components also have configurations; however, they constitute the *parametric configuration* part of the system configuration.

B. Diagnostics and Structural Reconfiguration

Structural reconfiguration by its nature depends on a *diagnostic image*, built from the measurements of the sensors deployed in the system. Diagnostic models can be ones that describe the knowledge of faults, errors and failures in the classic sense; however, the term is used here in an extended sense, as

reconfiguration can also be governed e.g. by a performance or a security model. The simplest type of diagnostic images is a vector of state attributes defined on a subset of the immutable components of the system; that is, components whose state is the diagnostic information are not added or removed. Binary diagnosis of hardware components (OK/FAILED) is generally of this type. The diagnostic image influences reconfiguration in two ways.

First, it can be an initiator: the current knowledge of the dynamic properties of the system is what drives the decisions whether a reconfiguration is needed and what are the problems in the structure that have to be dealt with, as failed, compromised or overloaded components. Second, it is a source of constraints on the target configuration; the goal of reconfiguration is to eliminate or avoid the effect of the known issues, identified by the diagnostic image. Thus, these are not to be reproduced; e.g. the target configuration must not allocate services to a physically failed server.

The need for a diagnostic image makes maximum speed and minimal-invasiveness of reconfiguration contradicting requirements, as fine granular diagnosis takes more time than rough, high-level ones.

C. Modeling Reconfiguration in the Engineering Domain

Configuration metamodels usually can be extended to incorporate diagnostic attributes and relationships (for example fault/error/failure modes, error propagation paths). A reconfiguration rule is the IT management workflow to be enforced under certain diagnostic images, leading from one configuration to another. Accordingly, reconfiguration can be described *by the target configuration* (declaratively) or *as a process* (imperatively), both in terms of the pre-established extended configuration metamodel.

Reconfiguration description by source and target configuration pairs, although needs additional workflow synthesis, offers the advantage of capturing diagnostic images in the same model. An aspect of declarative reconfiguration description currently under research is that a partial source configuration equipped with constraints is effectively a left hand side of a graph transformation rule on the configuration metamodel. This way, diagnostic image based reconfiguration initiation can be implemented as an incremental graph pattern search.

D. Structural Reconfiguration and Supervisory Systems

Traditionally, system management is realized in a strictly centralized manner with decisively human beings carrying out diagnosis and deciding on the actions to be taken. However, the IBM Autonomic Computing Initiative [3] and other efforts have been calling for 'closing the loop' with the automation of diagnosis and action planning since the beginning of the decade.

In structurally static IT architectures, design for Quality of Service takes system management into account mostly as a reactive repair process, acting against fault processes on a per component basis. Thus, system design and supervision/support design can be decoupled in an assume-guarantee manner via probabilistic component fault/failure and repair properties. In contrast, in a system where (semi-)automatic structural reconfiguration is a resilience mechanism, overall Quality of Service (QoS) is a derivative of the QoS properties of multiple configurations and the QoS shown during reconfiguration. Therefore, design for user-observable service quality must take not only the QoS of individual configurations, but also *Quality of Management (QoM)* aspects as e.g. switchover times or time to reconfiguration decision into account. These needs call for an integrated, model-based design process for environments managed with structural reconfiguration.

E. An Experimental Design Workflow for Virtualized Data Centers

Currently no unifying model driven design process template exists for the integrated design of configurations and reconfigurations. With the author as project co-supervisor, an experimental design workflow for a constrained problem was formulated and prototyped in [4]. This work focused on virtualized data centers with a fixed set of host machines, a set of virtual machine images and additional performance and dependability constraints, e.g. the minimal number of replicas to be provided for each image to express cluster size and standby needs.

A configuration metamodel – deployment of independent images on physical hosts with piecewise linear modeling of the host resource allowance - performance relations – and diagnostic metamodel (physical host failures) were formulated; models of these are transformed to a a mixed integer linear programming (MILP) problem for finding alternative, repairing configurations based on an initial configuration and its diagnostic image. Ilog CPLEX was used as an industry-strength MILP solver [5]. The user-defined optimization objectives may cover an arbitrary mix of goals like maximization of the availability of high priority services or minimizing the operational costs originating from resource usage. IBM Tivoli Provisioning Manager was used as an IT workflow automation platform. The resulting workflow is depicted on Figure 1. Currently, the logic of the workflow synthesis from the declarative solver results is quite ad-hoc and rudimentary; we are currently examining the possibilities for synthesizing the workflow during problem solving, as well.

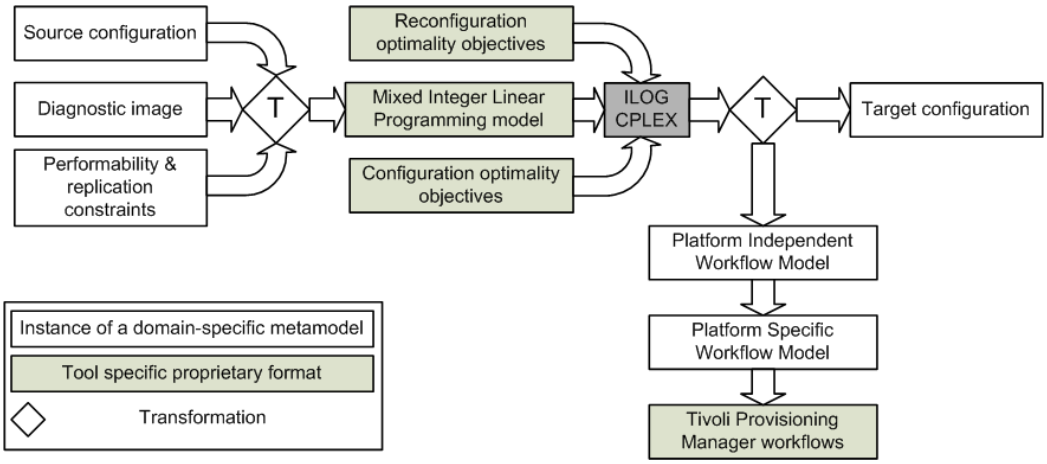

Figure 1: Overview of the experimental configuration and reconfiguration design workflow

III. Dependability and Quality Aspects of Reconfiguration

Model driven design enables for automatic verification of system properties of interest via hidden formal methods. For static infrastructures, service and system related metrics of various fields as performance, availability or safety engineering are well known. However, systems employing structural reconfiguration impose new challenges in terms of properties to be verified.

- Although the definition of service level properties remains the same, analytic modeling and mathematical solving needs to be modified to take into account the supervision-initiated reconfigurations.
- The system is in a transient state during reconfiguration. System dependability and performance can be relaxed during reconfiguration only to the level explicitly accepted as tolerable.
- Measures are needed to be able to express the 'operational range' of the system under the known influencing conditions, as they are used for structurally static dependability mechanisms (for example, number of faults tolerated).

In the following, some initial results are reported.

A. Uninterpreted Modeling and Analysis of Reconfiguration

Uninterpreted modeling of reconfiguration is modeling by using the domain-invariant concepts of the mechanism: configurations, reconfiguration workflows and diagnostic image initiated reconfigurations.

A system utilizing structural reconfiguration can be treated as a state machine, with configurations as states, reconfigurations as transitions and diagnostic images as guard conditions. In the following, we suppose that the components under diagnosis are immutable by the reconfiguration and that the

diagnostic vector is the same for all configurations. Among others, this formalism allows to check the completeness and the determinacy of reconfiguration rule sets.

Let us construct another, probabilistic state machine, where the state vector is the fault and failure state vector of the components modeled. Then let us extend this state vector with the diagnostic image, where the diagnostic infrastructure and diagnostic logic can be pessimistically approximated by deterministic, timed transitions. Also, let us extend the reconfiguration model with transition times. Let us derive the product automaton of these two automata, with its state vector extended with the (pessimistic) service failure modes that is a function of the actual dependability state and the current configuration. This automaton represents the uninterpreted managed behavior of the system in terms of faults and failures. This state machine is amenable to probabilistic analysis as well as model checking of the user observable service and the management logic.

B. Reconfigurability as a Quality of Management Measure

Intuitively, the 'more reconfigurable' a system is, the more possibilities system management has to handle incidents and environmental changes. Reconfigurability measures may aim at capturing:

- the number of configurations reachable from a given configuration on a per trigger event type basis (*reconfigurability of configurations*),
- process characteristics as for example execution time (*quality of reconfiguration*), or
- the (minimal/maximum/average) length of the reconfiguration series path from an initial configuration that leads to a configuration that has no reconfigurability directive for a condition that needs handling (*reconfiguration run lengths*).

Additionally, reconfigurability measures have three fundamentally different contexts.

- *Inherent reconfigurability measures* are those that an idealistic supervisory system would realize on a given environment. These act as upper bounds for implemented solutions and are defined by the available resources and the needs of the services to be implemented.
- *Supervision design reconfigurability measures* are those that characterize the configuration state space and its transitions determined in system and supervision design time. In practical cases, this configuration transition system is likely to be a small subset of all possible configurations and reconfigurations; however, to fully utilize the resilience possibilities inherent to the system, reconfiguration run lengths for typical trigger event sequences and quality of reconfiguration measures should be comparable with the inherent ones (provided that those can be estimated at least).
- *Managed reconfigurability measures* are interpreted on the reconfigurations enabled by the current internal diagnostic image of a supervisory system at any given point of time. Here, the main issue is that how well and how fast supervision converges to the designed measures.

Acknowledgement

The work reported on in the paper was partially supported by the European FP6 project DESEREC.

References

[1] W. Bumpus et al., *Common Information Model: Implementing the Object Model for Enterprise Management*, John Wiley & Sons, 2000.

[2] H. Madduri, S. Shi, R. Baker, N. Ayachitula, L. Shwartz, M. Surendra, C. Corley, M. Benantar, and S. Patel, "A configuration management database architecture in support of IBM Service Management," *IBM Systems Journal*, 46(3), 2007.

[3] A. Ganek and T. Corbi, "The dawning of the autonomic computing era," *IBM Systems Journal*, 42(1):5–18, 2003.

[4] I. Váncsa, "Construction of a virtualization-based high-availability server," Scientific Student Conference Report, BME, 2007.

[5] ILOG, "CPLEX 9.0 User Manual," 2004.