System Architecture, Dependability and Modes

Linas Laibinis, Elena Troubitsyna
Aabo Akademi
Turku, Finland
{linas.laibinis, etroubit}@abo.fi

Alexei Iliasov, Alexander Romanovsky
Newcastle University
Newcastle Upon Tyne, UK
{alexei.iliasov, alexander.romanovsky}@ncl.ac.uk

***Abstract*—The mode, defining the specific type of functional behaviour that a system exhibits during its operation, is an important architectural level concept, which has a significant impact on system design, verification and dependability. The notions of modes and mode changes are widely used by the industrial engineers to structure reasoning about different conditions of system functioning. Even though there has been some work on developing modal systems, we still lack a general understanding of how to architect, verify and ensure dependability of such systems. In our work we rely on formal modelling and verification to study intricate relationships between fault tolerance, operation modes and architectural design**.

Keywords-fault tolerance, formal methods, structuring

I. INTRODUCTION AND MOTIVATION

Operation modes [1, 2] form a useful structuring concept that facilitates the design of safety-critical systems in different industrial sectors, including avionic, automotive, transportation and space. Modes define functionality of the system in relation to its own state and the state of its environment. In particular, the concepts of modes and mode changes are widely used for structured reasoning about normal and abnormal behaviour of systems and their components, recovery, degradation and reconfiguration. Our experience in the FP7 ICT project DEPLOY (www.deploy-project.eu/) shows that many domain experts express system requirements and define system behaviour in terms of system and component modes and mode transitions [3].

There are several well-known problems associated with modal systems, e.g., ensuing common operation mode in distributed systems, verifying correctness of mode transitions, mode confusion etc. It is clear, that proper dealing with modes will have a direct impact on system dependability. Yet, there is still a need for generic architectural-level approaches to designing modal systems. These approaches should support hierarchical system structuring using modes and components. The developers should be able to model the local modes of individual components as well as the global modes of the system in a systematic way. This will help them to alleviate the above-mentioned problems inherent to modal systems and to enhance dependability.

In our work we develop formal approaches to architecting, designing and verifying complex embedded systems. We advocate the use of the hierarchical architectures and modularized formal development by refinement as the main techniques for mastering complexity of modal systems and assuring their dependability.

II. SOFTWARE ARCHITECTURE AND MODAL SYSTEMS

It is widely recognized that a layered architecture is preferable in designing complex control systems since it allows developers to map real-world domains into software layers. Usually the lowest level confines the embedded real-time subsystems which directly communicate with sensors and actuators. These subsystems cyclically execute the standard control loop consisting of reading the sensors and assigning new values to the actuators. The layer above contains the components that encapsulate the detailed behaviour of the lowest level subsystems by providing abstract interfaces to them. At the highest level of the hierarchy there is a mode manager - the component responsible for system mode transition.

We assume that the overall system should execute a certain autonomous scenario defined in terms of modes (a well-known example is a flight scenario, with the take off, ascending, cruising, descending, landing modes). Essentially each mode is characterized by a certain set of component states. Transition between the modes is guided by this scenario as well as by the states of the system environment and components. The mode manager monitors these states. When it detects that the conditions required for entering the next mode are satisfied, it commands the lower layer components to set their new operation modes accordingly. Upon receiving a mode changing request, the lower layer components perform the similar type of calculations and issue the corresponding requests further down in the hierarchy.

However, due to component failures or certain environment conditions the system might need to deviate from the consecutive execution of the steps of the predefined scenario. For instance, a plain might need to ascend at the landing mode and redo landing due to air traffic or weather conditions at the airport or internal aircraft problems. Hence the mode manager should be also to detect failure conditions and properly execute certain backtracking mode transitions in a correct way as well.

Obviously, incorrect handling of mode transitions might have major impact on system dependability. Unfortunately, verification of correctness of mode transitions is complicated by possibility of multiple component failures, changing operational conditions or failures of other components during error recovery. To deal with the problems of these types we propose a formal approach to architectural level design and refinement of complex modal systems.

III. Architecting Component Modes and Mode Transitions

Our previous work [2, 4] presents the formal definitions of the abstractions used to specify modal systems. According to this approach, an architecture of a modal system is an abstract specification of the modes as well as mode transitions that may occur in a system. It specifies neither how the system operates while it is in some specific mode nor how mode transitions occur. It rather imposes restrictions on concrete implementations, complementing traditional modelling but not replacing it. These ideas have been further applied for architecting fault tolerance modes [2].

Currently we are expanding this approach by showing how to derive the layered architecture of the system while preserving correctness of mode transitions. In this architecture the modes of components are visible at the component interfaces and can be accessed by the higher-level components. We define a generic modelling pattern and specify each component by instantiating it. The overall system architecture is built by recursive application of this pattern.

We use Event-B [5], a state-based formalism closely related to Classical B [6], to formally model both the overall system architecture and system components. Event-B is a state-based framework that suits well to modelling and verification of modal systems. Proofs that accompany the development allow us to formally verify mode invariants, correctness of mode transitions as well as consistency between system and component modes.

We are now developing Mode and Modularization plug-ins supporting this approach as part of the Eclipse Rodin environment (www.event-b.org/). The Mode plug-in, providing a modal (orthogonal) view on the system architecture, will allow the architect to graphically capture system modes and mode transitions.

IV. Discussion

Currently we are validating our approach by modelling the architecture of the Attitude & Orbit Control System (AOCS) [7]. AOCS is a typical representative of a class of control systems. Its main function is to control the attitude and the orbit of satellites. The system consists of the AOCS Mode and FDIR (Fault Detection, Isolation and Recovery) Manager, the Unit Manager and seven instruments. Various mission stages define the autonomous scenario that the mode manager should enforce. Each mode involves specific functions performed by different instruments. The Mode Manager controls the AOCS operation as specified by its modes and their transitions. Each mode transition is described as a set of specific actions to be performed in order for the mode to get changed. Transitions between modes occur either because conditions for entering the next mode in the autonomous scenario are reached or because a failure has occurred and the system needs to backtrack.

In developing the AOCS system we aim to validate the proposed specification and refinement patterns as well as to explore the scalability of the proposed approach to reasoning about complex mode transitions. Our initial experience shows that our approach supports well correct and consistent decomposition of the global system modes into modes of individual components and allows the system architect to prove the consistency of mode changes while developing a system architecture by refinement.

In our future work will plan to integrate architecting the tasking structure into the models using refinement patterns, as well as reasoning about system schedulability.

There has been some work on architecting systems using the concept of operation modes. In the Architecture Analysis & Design Language (AADL) [8] a system is built out of communicating components and each component may have modes, representing alternative operational states. AADL modes identify configurations of components. A state machine abstraction is used, such that a distinct configuration is a modal state and specific events cause transition among them. A component may have distinct behaviour according to the current mode. An AADL components ican be recursively structured as a number of interlinked subcomponents. In [9] the authors purpose an architectural support for mode-driven fault tolerance in which the mechanisms for tolerating hardware faults are associated with the modes and the mode transitions are specified as the finite state machines. In [10], the representation of degraded service outcomes and exceptional modes of operation using UML use cases, activity diagrams and state charts are discussed. Unfortunately neither these nor other papers know to us support rigorous reasoning about mode and mode transitions by refinement, providing at the same time a rich set of abstractions to recursively reason about system and component modes.

References

[1] F. Jahanian, A. Mok, Modechart, “A specification language for real-time systems,” IEEE Transactions on Software Engineering, vol. 20, no. 12, pp. 933–947, 1994.

[2] A. Iliasov, F. Dotti, A. Romanovsky, “Structuring Specifications with Modes.,” Proc. Fourth Latin-American Symposium on Dependable Computing (LADC), September 1-4, 2009, Brazil. IEEE CS. 2009.

[3] DEPLOY Deliverable D5 - Report on knowledge transfer. FP7 ICT DEPLOY Project. Jan 2009. www.deploy-project.eu.

[4] F. Dotti, A. Iliasov, L. Riberiro, A. Romanovsky, “Modal Systems: Specification, Refinement and Realisation,” Proc. Conference on Formal Engineering Methods - ICFEM 09, December 9 -12, 2009, Rio de Janeiro, Brazil. Springer. 2009.

[5] J. R. Abrial, C. Metayer, “RODIN deliverable 3.2 - Event-B language,” ICT RODIN Project Deliverable. Newcastle University, UK. 2005. http://rodin.cs.ncl.ac.uk.

[6] J. R. Abrial. The B-Book: Assigning Programs to Meanings. Cambridge University Press. 2005.

[7] DEPLOY Deliverable D20 - Report on Pilot Deployment in the Space Sector. FP7 ICT DEPLOY Project. Jan 2010. www.deploy-project.eu.

[8] P. H. Feiler, D. P. Gluch, J. J. Hudak, “The architecture analysis & design language (AADL): An introduction,” TN-011, SEI, 2006.

[9] D. Srivastava, P. Narasimhan, “Architectural support for mode-driven fault tolerance in distributed applications,” SIGSOFT Softw. Eng. Notes, vol. 30, no. 4, pp. 1–7, 2005.

[10] S. Mustafiz, J. Kienzle, A. Berlizev, “Addressing degraded service outcomes and exceptional modes of operation in behavioural models,” Proc. International Workshop on Software Engineering for Resilient Systems (SERENE). NY, USA: ACM, 2008, pp. 19–28.