Modelling and Monitoring NFRs in Autonomic Systems: AS-TROM Approach

O. Ormandjieva

Abstract – **This paper is the first report on the ongoing research targeting the rigorous development of autonomic (self-managing) systems with built-in continuous monitoring of their non-functional requirements for quality through self-diagnosis, followed by planning. The research is focusing on Autonomic Systems - a significant and new strategic and holistic approach to the design of computer-based systems. Autonomic elements have complex life cycles, continually sensing and responding to the environment in which they are functioning. Therefore, the autonomic system can be classified as a real-time reactive system. The autonomic system requires solid formal foundations for system development and functioning. The Timed Reactive Object Model (TROM) formalism for real-time reactive systems, created at Concordia University, is being extended to model autonomic systems whose architecture, system configuration, and continuous self-monitoring of their quality are to be specified within a single formal framework.**

Keywords – **Autonomic Systems, Formal Methods, Software System Quality, Non-Functional Requirements (NFRs).**

I. INTRODUCTION

Software systems are characterized both by their functionality (what the system does) and by their non-functionality (how the system behave with respect to some observable attributes like reliability, reusability, maintainability, etc.) Both aspects are relevant to software development. However, non-functional issues have received little attention compared to functional ones. The non-functionality is addressed by just a few approaches, often semi-formal or informal and limited in scope. The increasing trend toward developing complex software systems has highlighted the need to build software non-functional requirements (NFRs) into the software system. To model and validate these non-functional requirements new techniques have to be developed in addition to existing formal methods and tools.

This paper is focusing on Autonomic Systems - a significant and new strategic and holistic approach to the design of computer-based systems. This is a new and challenging area in Software Engineering discipline emerged in 2001 from the needs of the industry [6, 7] that has created interests in different research groups worldwide.

The main characteristic of autonomic computing is self-management, i.e., continually monitoring of its own use and quality in the face of changing configurations and external conditions based on automatic problem-determination algorithms. One of the most important aspects of self-management is to perform self-diagnosis to check the system's quality status. Building self-monitoring system requires specifying what to monitor. In our approach, a set of non-functional requirements of quality expressed as constraints on the functional requirements, forms the set of rules for monitoring.

The automation of system self-management requires solid formal foundations for system development, including integration of the NFRs into the development process. The Timed Reactive Object Model (TROM) formalism created at Concordia University [10, 11] has the required expressiveness power for specifying autonomic elements. One of the first objective of this research is to extend TROM formalism to include the specification of the autonomic system architecture, configuration, NFRs and self-monitoring rules within a single formal framework AS-TROM.

The paper is organized as follows: Section 2 describes the AS-TROM formalism. Section 3 introduces the NFRs in Software Engineering. The related work is surveyed in Section 4. Our approach is explained in Section 5. The conclusions and the future work directions are outlined in Section 6.

II. FORMALISM

The TROMLAB development environment is an integrated facility based on the TROM formalism [10] for modeling, analyzing, and developing real-time reactive systems. The process model in TROMLAB supports the iterative development approach, which provides the following benefits:

- Reduces risks by exposing them early in the development process.
- Gives importance to the architecture of the system's configuration.
- Designs modules for large-scale software reuses.

The TROM formalism is a three-tier formal model [10]. As a layered model, each upper tier communicates only with its immediate lower tier. The independence between the tiers makes the modularity, reuse, encapsulation, and hierarchical decomposition possible. The three-tier structure describes the system configuration, reactive classes, and relative Abstract Data Types. The upper-most tier is the subsystem configuration specification. It specifies the object definition, their collaboration, and the port links, which regulate the communication tunnels between objects. The middle tier is the TROM class, which is a Generic Reactive Class and is included in the subsystem. TROM class is a hierarchical finite state machine augmented with ports, attributes, logical assertions on the attributes, and time constraints. The lowest tier is the Larch Shared Language (LSL) trait that represents Abstract Data Type used in the TROM classes.

The AS-TROM formalism is extending the TROM formalism through adding one more tier (see Fig. 1) to include the specification of the autonomic system architecture, system

configuration, and self-monitoring within a single formal framework. AS-TROM is expressive enough for developing autonomic elements.

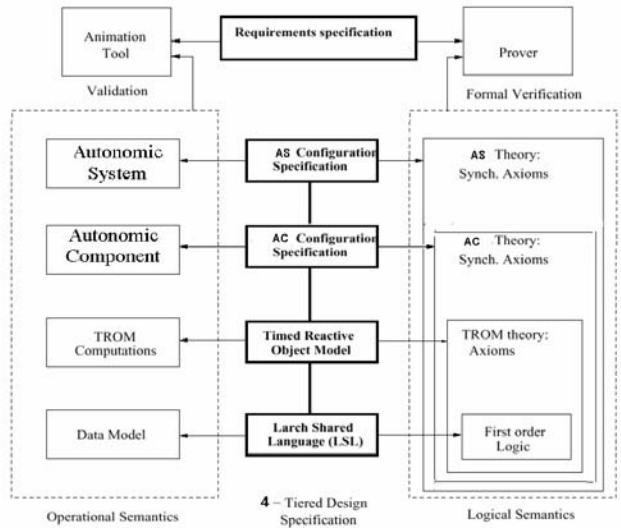

Fig. 1. AS-TROM: four-tier formal model

The design of the autonomic system is specified through formally modeling the autonomic components and their relations, and the timing requirements constraining system's behavior to ensure safety and liveness properties of the system. The formal model of the autonomic system design has to be validated by simulating its behavior and reasoning on the results from the simulation. System verification takes place at the next stage. The AS-TROM model of the system has to be mechanically translated to a set of PVS [9] theories consisting of axioms describing the timed behavior of the system. Time critical properties such as safety and liveness, are to be included as lemmas in PVS theories and verified formally similarly to the current TROM process (see Fig 2).

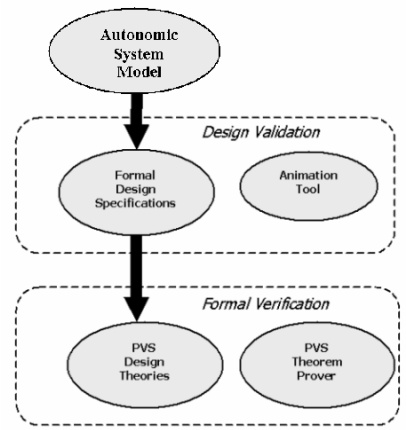

Fig. 2. AS-TROM: Specification, Validation and Verification Methodology

III. NFRs

Once a software system has been deployed, it is straightforward to observe whether or not a certain functional requirement has been met, as the areas of success or failure are rigidly defined. The same is not true for NFRs of quality as these refer to measurable quantities, which usually tend to be strongly interdependent and are among the most expensive and difficult to deal with [1, 2]. According to the software engineering standard IEEE Std.830-1998 [12], NFR is defined as "a software requirement that describes not what the software will do, but how the software will do it, for example, software performance requirements, software design constraints, software external interface requirements and software quality attributes". NFRs had been neglected by the requirements engineering practice and research. Usually NFRs are expressed in a natural language, suffering from ambiguities and potential conflicts. As a consequence, NFRs are difficult to validate and verify; therefore, they are usually evaluated subjectively. The above, and the importance of the NFRs for developing high quality complex systems have motivated the second objective of this work, namely, to develop the hierarchical model of quality NFRs and integrate its specification within the AS-TROM formalism through mapping to system's functional requirements specification.

IV. Related Work

The most widespread approach for dealing with non-functional requirements is the NFR framework [1]. A very important aspect of non-functional requirements decomposition using the NFR Framework is that, as far as NFR softgoal are refined into more detailed ones, it is possible to identify interactions between non-functional requirements. These interactions include positive and negative contributions and have a critical impact on the decision process for achieving other non-functional requirements.

The Unified Process for developing OO systems provides a relatively minimal level of support for expressing NFRs [2]. Extension mechanisms of the UML standard [8] have been used to capture the non-functional requirements expressed in NoFun [4, 5], a language created to provide a basis for establishing quality models in a formal way. The work targets the complete set of quality requirements as described in the ISO/IEC 9126 International Standard. The authors propose the separation of FRs and NFRs where each class is associated with an NFR element, expressed in OCL [5].

Paper [3] discusses a sequence of systematic steps towards an early consideration of specifying and separating requirements. This makes it possible to identify and resolve conflicts earlier in the development cycle and promotes traceability of broadly scoped requirements throughout system development, maintenance and evolution. The approach is presented within four categories of activities: FRs identification and specification, NFRs identification and specification, composing requirements and analysis/design activities. Formalism for the FRs is provided through specifying their pre and post conditions formally using first order predicate logic.

Our work builds upon the existing methods but differs from them in important ways: i) this research proposal takes

advantage of the formal representation of components in AS-TROM formalism, and the autonomy of components in agent-oriented paradigm; ii) formalization of both FRs and NFRs within the same formal framework AS-TROM so that the NFRs can be validated and verified automatically.

V. APPROACH

Autonomic systems automatically monitor and seek opportunities to improve their own quality characteristics such as reliability, availability and performance. The corresponding NFRs have to be specified formally and mapped to autonomic elements' behaviour so that the achievement of the above NFRs can be monitored automatically. Behavioural changes due to the environment and/or system evolution have to be detected automatically, and the self-diagnosis of the system quality against changes must be modelled, followed by the planning of self-healing reaction when the system's behaviour fails to meet the NFRs requirements. The changed configuration may be verified while the system is running without affecting the system integrity. We are currently working on the development of a new AS-TROM formal language powerful enough to describe the structure and the behaviour of the autonomic systems, as well as the non-functional properties constraining the behaviour of the system. The development process formalization would allow for formal validation and verification of FRs and their conformance to the corresponding, guarantying the high quality of the final product and allowing for continuous quality control on the evolving software structures.

The automation of system self-management requires solid formal foundations for system development, validation and verification, including integration of the NFRs into the system formal specification. In this research, each NFRs is regarded as a mathematical theory whose set of axioms is defined by the required scale type of the measurement, and the empirical observations on the attribute to measure. NFRs in this context are regarded as abstractions of algorithms for their quantification, without imposing restrictions on the measurement mechanism other than the axioms specified in the theory. TROM formalism allows for theory inclusion, therefore, the NFRs constraining an autonomic element are to be included in the theory of the corresponding element's specification, thus formalizing the mapping of non-functional requirements to the specification of the components' behavior. Formally expressed NFRs within the same AS-TROM language are formalized as theorems and should be provable from the autonomic elements' behavioral specification automatically.

An important step in achieving successful NFRs monitoring is to achieve the right balance of system quality attributes. The quality model is a rigorous hierarchical decomposition of NFRs for quality from their general statements to algorithms for their evaluation (measurements). This involves identifying the conflicts among several desired quality attributes, and working out a satisfactory balance of attributes satisfaction. The quality model for autonomic systems chosen for our work is the one developed for real-time reactive systems, and based on the TROM formalism as described in [11]. The TROM quality model differs from the existing quality models for reactive systems: i) the measurements are theoretically validated, and ii) are based on the formal specification of the system in TROM. Therefore, it allows the assessement of quality of design solutions at early in the development process.

The solid formal bases for the measurements specified within the same formal framework allow for automation of the evolving automatic system's self-monitoring mechanism. The research achievements in modelling of two of the most important autonomic systems quality requirements, namely, reliability and performance, are described below.

Reliability. The reliability assessment model based on TROM formalism and its empirical validation has been reported in [13, 14, 15, 16, 17, 18]. The reliability model based on Markov chains of the system's expected change is constructed from the system configuration specification. This model serves as the basis of the evolution engine that calculates the reliability prediction factors, and will be used for formalizing the autonomic systems' early reliability assessment. The reliability requirements for autonomic elements and systems have to be specified formally and mapped to system behaviour so that the achievement of the reliability can be monitored automatically.

Performance. The current research work includes performance formalization within the same framework and their crosscutting with autonomic system's behaviour. Some of the results have been published in [19, 20].

Novelty. To the best knowledge of the authors, there is no similar approach reported in the literature on formal specification of non-functional requirements and the system functional model within the same formal framework, for autonomic systems.

VI. DISCUSSION AND NEXT STEPS

This paper is a preliminary report on the ongoing research which main objective is the rigorous approach to developing and evolving autonomic systems whose architecture, system configuration, and continuous self-monitoring of their quality are to be specified within a single formal framework.

The research is focusing on Autonomic Systems - a significant and new strategic and holistic approach to the design of computer-based systems. Autonomic systems will need a mechanism to acquire and represent high-level specifications of NFRs and map them onto lower-level actions. We must develop and analyze algorithms and negotiation protocols for conflicting NFRs, and determine what bidding or negotiation algorithms are most effective.

Safety and Liveness. One of the most challenging tasks in autonomic software system self-monitoring is to assure the conformance to the safety and liveness properties, especially as these systems are to be used in sensitive and often life-critical environments such as medical systems, air traffic control, and space applications. In AS-TROM these properties will be expressed as invariants on the system's behavior, that is, it should be possible to derive those properties from the postcondition of each the system function leading to a change of state. The formalization of the safety and liveness

properties would allow for their automatic monitoring after each state change of the system.

Mechanical Support. The AS-TROM process formalization would allow for formal validation and verification of FRs and their conformance to the corresponding NFRs, guarantying the high quality of the final product and allowing for continuous quality control on the evolving software structures. The tool chosen for verification purposes is PVS [9], a verification assistant that provides mechanized support for formal specification and verification and is based on classical, typed higher-order logic. A mechanism for mapping AS-TROM specifications to PVS theories has to be developed as part of this research work.

Long term objectives. The basic issue for autonomic systems will be to combine the processing of environment and machines to create a more effective overall computation. The interaction between the environment and the autonomic systems will be framed by the predictability and trustability issues, which depend on the unexpected system behavior. The long term work in this direction shall be contributing to building up of a shared ground for environment–computer interaction.

REFERENCES

[1] L. Chung, B. A. Nixon, E. Yu and J. Mylopoulos, "Non-Functional Requirements in Software Engineering". *Kluwer Academic Publishing*. 2000.

[2] R. R. Yong. "The Requirements Engineering Handbook". *Artech House Publishers*. 2004.

[3] M. Kassab, C. Constantinides, O. Ormandjieva. "Specifying an

Separating Concerns From Requirements to Design: A Case Study". *Accepted at ACIT-SE 2005*.

[4] Franch X., Botella P. "Putting non-functional requirements into software architecture". *In Proceedings of the Ninth International Workshop on Software Specification and Design*, pp. 60– 67, 1998**.**

[5] Pere Botella, Xavier Burgues, Xavier Franch, Mario Huerta, Guadalupe Salazar. "Modelling Non-Functional Requirements", 2001.

[6] P. Horn. "Autonomic Computing: IBM's Perspective on the State of Information Technology". *IBM Manifesto*, October 2001, http://researchweb.watson.ibm.com/autonomic/manifesto

[7] J.O. Kephart, D. M. Chess. "The Vision of Autonomic Computing". *IEEE Computer*, January 2003, pp.41-50.

[8] M. Fowler. "UML Distilled 3/e: A Brief Guide to the Standard Object Modeling Language". *Addison-Wesley Pearson Education, Inc*. 2004.

[9] PVS: http://pvs.csl.sri.com/

[10] V.S. Alagar, R. Achuthan, D. Muthiayen, "TROMLAB: A Software Development Environment for Real-Time Reactive Systems", *Technical Report, (first version 1996, revised 1998)*, Concordia University, Montreal, Canada.

[11] O. Ormandjieva. "Deriving New Measurement for Real Time Reactive Systems". *Ph.D. Thesis*, Computer Science & Software Engineering Department, Concordia University, 2002.

[12] IEEE Std.830-1998. "Software Requirements Document".

[13] V.S. Alagar, O. Ormandjieva. "Reliability Assessment of E-Commerce Applications". *In Proceedings of 1st International Conference on E-Business and Telecommunication Networks (ICETE 2004)*, pp.30-37.

[14] V.S. Alagar, O. Ormandjieva, M. Zheng. "Two-tier Agent Architecture for Trusted Communication in Ad-Hoc Mobile Networks (Extended Abstract)". In *Proceedings of the Winter International Symposium on Information and Communication Technologies 2004 (WISICT04)*, pp.392-397.

[15] V.S. Alagar, M. Haydar, O. Ormandjieva, M. Zheng. "A Rigorous Approach for Constructing Self-Evolving Real-Time Reactive Systems". *Journal of Information and Software Technology, (2003*), pp. 743-761.

[16] V.S. Alagar, O. Ormandjieva, M. Zheng. "Incremental Testing for Self-Evolving Timed Systems". In *Proceedings of the Third International Conference on Quality Software (QSIC 2003)*, p.12-19.

[17] V.S. Alagar, O. Ormandjieva,M. Zheng. "Managing Complexity in Real-Time Reactive Systems". In *Proceedings of the Sixth IEEE International Conference on Engineering of Complex Computer Systems (ICECCS2000)*, Tokyo, Japan, 2000.

[18] V.S. Alagar, Ormandjieva. "Reliability Assessment of Web Applications". *In Proceedings of the 26th Annual International Computer Software and Applications Conference (COMPSAC 2002)*, pp.405-414.

[19] V.S. Alagar, O. Ormandjieva, Shi Hui Liu. "Scenario-Based Performance Modelling and Validation in Real-Time Reactive Systems". *In Proceedings of Software Measurement European Forum 2004 (SMEF2004)*.

[20] V.S. Alagar, Shi Hui Liu, O. Ormandjieva, Jian Shen. "Performance Assessment in Real-Time Reactive Systems". *In Proceedings of the 7th IASTED International Conference on Software Engineering and Applications (IASTED - SEA 2003)*, pp.397-206.