**A Systematic Review of Runtime Test Case Synchronization in Adaptive Software**

Mohammed Akour

Department of Computer Science

North Dakota State University

Fargo, ND 58108, USA

mohammed.akour@ndsu.edu

Gursimran S. Walia

Department of Computer Science

North Dakota State University

Fargo, ND 58108, USA

gursimran.walia@ndsu.edu

Tariq M. King

Department of Computer Science

North Dakota State University

Fargo, ND 58108, USA

tariq.king@ndsu.edu

**ABSTRACT**

The quality of software system continues to be an important and challenging issue in software engineering. A software system will need to upgrade or reconfigure to satisfy the proposed qualities and capabilities. The changes can occur at development time or at runtime. The changes in runtime conditions are relevant for autonomous (dynamic adaptable) systems. A software system is composed of a set of elements, which are related to each other; and changing one element may affect the work of others which are related to it. The testing model should be updated to be consistent with the changes. The goal of our research is to investigate how Model driven development approaches such as change propagation can be harnessed to maintain consistency and completeness between models of the runtime configuration of adaptive software and their associated test models.

# INTRODUCTION

The systems, technologies, and enterprises of today have become highly complex and heterogeneous. This complexity continues to increase as functions are added, removed and modified dynamically to meet the growing needs of customers, to handle exceptional phenomena, and to adapt the system to new operating environments. Computing systems and infrastructures have therefore become more difficult to manage, maintain, and integrate. Self-adaptive software paradigms such as autonomic computing [6] have been developed with the goal of shifting the low-level details of software operation and maintenance from administrators to technology. The idea is to allow administrators to specify system goals as high-level policies, and have the system adapt its own components in response to changes in the operating environment [6, 7]. By operating environment, we mean those things that are externally observable by the software system such as the end users, operating system, file system, and other software applications and libraries that the system needs [1].

At runtime, adaptive software continually monitors its environment and its own runtime operations, and determines whether or not upgrades or dynamic re-configurations are necessary [10]. These changes are then automatically realized through the use of a dedicated self-management infrastructure. In essence, the advent of adaptive and autonomic software has moved traditional software maintenance activities such as repairing faults and adding new functionality into the runtime environment. However, unlike traditional software maintenance activities, validation or verification of changes made to the adaptive software after initial deployment has not been emphasized as part of their conception, promotion, and development.

Researchers have started to recognize the need for runtime validation and verification to be an inherent part of dynamically adaptive software [3, 8, 9]. However, although software testing remains the primary means of validation used in industry, there are few works on testing adaptive software at runtime [10].

King et al. [3] propose the use of an integrated autonomic self-testing framework to validate self-adaptive behaviors at runtime. Their framework seeks to ensure that: (1) new errors have not been introduced into previously tested components through on-line regression testing, and (2) validate the behavior of added functionality by serving as the focal point for dynamic test case generation and execution. They also recognize the need for reduction of the test set when components are removed from the system configuration. However, their description of autonomic self-testing does not provide solutions or insights into the problem of maintaining up-to-date test suites in adaptive software at runtime.

Model-driven development (MDD) [4] has focused on instating models as first-class citizens during the software development process. As such, model transformations and synchronization are key goals of MDD. However, many approaches only allow one shot transformations to be expressed [5]. Change propagating model transformations are those which can make suitable updates to models after an initial transformation [5]. Stated formally, the goal of this research paper is to:

*Analyze* Synchronization approaches

*for the purpose of* harnessing change propagation

*with respect to* their consistency and the completeness between models of the run time configuration

*from the point of view of* researcher

*in the context of* run time synchronization of tests for adaptive software.

As a preliminary step to our investigation, we performed a systematic literature review to survey the research landscape surrounding test case synchronization in adaptive software. A systematic review brings together information from many studies, but provides something more than a range of all conclusions to date or a consensus of the views of the most prolific or prominent authors. It aims to synthesis the smaller studies, taking account of their size and quality to produce a single answer to a specific question. The findings of our literature review analyzed whether or not change propagation represents a formidable direction towards a solution to the problem of runtime synchronization of tests for adaptive software. The rest of this paper is organized as follows: Section 2 contains background information… Section 3 presents details on the systematic review process. Section 4 and 5 reports the results of the review. Finally, the conclusions and findings are discussed in Section 5.

# Background

To provide context for the review, we will first describe background material on the problem under investigation. This includes information on software maintenance, and software evolution. Software maintenance involves making changes to the software after delivery in order to achieve different targets. Four categories of maintenance includes: 1) Corrective maintenance in which the changes are made to debug the system after a failure has been observed, 2) Adaptive maintenance in which changes are made to achieve continuing compatibility with the target environment or other systems, 3) Perfective maintenance in which the changes are made to improve or add capabilities, and 4) Preventative maintenance in which the changes are made to increase robustness, maintainability, portability and other characteristics **.** The critical issue is that adaptive software should deal with the dynamic/run time changes which include the traditional standards of maintenance, not only the static ones.

Software development encompasses set of phases include planning, requirements, design, implementation, testing, and maintenance. Maintenance is the last stage of the software life cycle. After the products have been released, the maintenance phase keeps the software up to date with environment. While software development and software maintenance are two distinct and widely separated activities they are actually part of the same life cycle and the same importance needs to be assigned to both for overall success and good project management [2].

Evolution is crucial process in adaptive software development, so the systematic treatment of evolution is very important, At runtime, adaptive software continually monitors its environment and its own runtime operations, and determines whether or not upgrades or dynamic re-configurations are necessary, the software testers have started to recognize the need for runtime validation and verification provided by test models. However how to propagate modifications across models in the run time still remains as an open problem. Each element in adaptive model is related to at least one element in test model, we need to find appropriate approach to propagate that modification in adaptive model to test model to improve consistency and completeness between the models. Change propagation is central aspect of testing adaptive software at runtime, as adaptive system modifies one element by adding, updating or deleting, it must ensure that related test cases in test model are updated to be consistent with these new changes.

# Research Method

Following published guidelines, this review included the following steps

1. Formulate a review protocol.
2. Conduct the review (identify primary studies, evaluate those studies, extract data, and synthesize data to produce a concrete result).
3. Analyze the results.
4. Report the results.
5. Discuss the findings.

The review protocol specified the questions to be addressed, the databases to be searched and the methods to be used to identify, assemble, and assess the evidence. To reduce researcher bias, the protocol, described in the remainder of this section, was developed by one author, reviewed by the other two authors and then finalized through discussion, review, and iteration among the authors and their research group.

## Research Questions

The main goal of this systematic review is to analyze how the change propagation method represents a formidable direction towards a solution to the problem of runtime synchronization of tests for adaptive software.

To properly focus the review, a set of research questions were needed. With the underlying goal of providing support to the adaptive software system development, the high-level question addressed by this review was:

*Are there any approaches in the literature that can automatically synchronize the runtime test model of a software system with its component configuration in the response to dynamic software adaptations?*

This high-level question was then decomposed into the more specific research questions and sub-questions shown in Table 1. The first research question tries to gather evidence on usefulness of different approaches for synchronization between two models and in particular, synchronization between adaptive software model and their test models, finally to represent the limitation of the previous works in term of run time overhead of updating models. The second research questions tries to gather evidence on the use of change propagation as an approach to improve the consistency and completeness of adaptive software during the runtime. The final question analyzes Different change propagation languages were used to determine the possibility of using them to synchronize adaptive models with their test models.

**Table 1 – Research Questions and Motivation**

|  |  |
| --- | --- |
| Research Questions | Motivation |
| 1. Are there approaches in literature that focus on maintaining up-to-date models at runtime?
	1. Are any of these approaches applied in the context of adaptive software?
	2. Are any of these approaches applied in the context of updating test models?
 | Identify works related to the idea of synchronizing test models at runtime in adaptive software. |
| 1. Is there any evidence in the literature that multi-shot transformation approaches such as change propagation are effective for synchronizing different software models at runtime

 2.1 Does any of the evidence demonstrate that such approaches are useful for ensuring completeness and consistency of runtime models in software? | Assess usefulness of approaches in the literature for synchronizing runtime models without having to completely re-construct the target model. |
| 1. Are there any modeling tools, frameworks, or languages to support implementing approaches that synchronize runtime models
2. Are there any prototypes or case study applications that were built using these tools?
 | Assess practicality of developing a prototype of a solution to our specific problem using the approaches from (2.) |

## Source Selection and Search

Prior to conducting the search, the correct set of databases must be selected to optimize the likelihood of finding the most complete and relevant sources. In this review, the following criteria were used to select the source databases:

1. The databases were chosen to include journals and conference proceedings that cover: adaptive software system, change propagation models, model driven development.
2. The databases had to have a search engine with an advanced search mechanism that allowed keyword searches;
3. The list of databases was reduced where possible to minimize the redundancy of journals and proceedings across databases.

Based on the criteria for selecting database sources the final source list is shown in Table 2.

**Table 2 – Source List**

|  |  |  |
| --- | --- | --- |
| Databases | Other journals and Conferences | Additional Sources |
| IEEExploreACM LibraryScience DirectEBSCO | International Conference on Software Maintenance(ICSM) | Reference list from primary studies;Books  |

To search these databases, a set of search strings was created for each research question based on keywords extracted from the research questions and augmented with synonyms. In developing the keyword strings to use when searching the source databases, the following principles were applied:

* The major terms were extracted from the review questions and augmented with other terms relevant to the research;
* A list of meaningful synonyms, and alternate spellings was then generated. This list also included additional terms from papers that were known to be relevant;

The following global search string was constructed containing all of the relevant keywords and their synonyms:

*(Model OR software OR system OR development OR approach OR mechanism) AND (adaptive OR self adaptive OR adaptable OR adjustable OR autonomic) AND (configuration OR instrumentation OR installation) AND (completeness OR entirety OR integrity OR perfection OR fullness OR wholeness) AND (consistency OR agreement OR harmony OR correspondence OR accordance) AND (changing OR modification OR revision OR alternation OR updating OR maintaining ) AND (propagation OR spreading OR distributing OR broadcasting) AND (run-time OR run time OR dynamic OR operational-time OR running OR active-time OR real-time) AND (synchronous OR concurrence OR simultaneous OR coincidence OR configuration OR traceable) AND (limitation OR challenges OR threats OR dangers OR risks OR hazard )*

Using this global search string, five different search strings (each one with its own purpose) were derived and executed on each database. These strings are explained in Table 7 (see Appendix A). Executing the search strings on the databases in Table 2 resulted in an extensive list of potential papers that could be included in the review. To ensure that only the most relevant papers were included, a set of detailed inclusion and exclusion criteria was defined (Table 3).

Performing the searches from Table 3 on the databases in Table 2 resulted in an extensive list of potential papers that could be included in the review. To ensure that only the most relevant papers were included, a set of detailed inclusion and exclusion criteria was defined (Table 4).

Using these criteria, the results of the database searches were examined to arrive at the final list of papers. The process followed for paring down the search results was:

1. Use the *title* to eliminate any papers clearly not related to the research focus
2. Use the *abstract* and *keywords* to exclude additional papers not related to the research focus
3. Read the remaining papers and eliminate any that are not related to the research questions.

This processes resulted in a list of papers to include in the systematic review. After using the inclusion and exclusion criterion to select relevant papers, a quality assessment was performed on those papers. This quality assessment was another check on the quality of papers that resulted from the initial search.

Each accepted study after using the inclusion and exclusion criterion and removing duplicated studies is assessed for its quality against set of criteria. Some of these criteria were informed by those proposed for the Critical Appraisal Skills Programme (CASP) (in particular, those for assessing the quality of qualitative research) and by principles of good practice for conducting empirical research in software engineering.

**Table 4 – Inclusion and Exclusion Criteria**

|  |  |
| --- | --- |
| Inclusion Criteria | Exclusion Criteria |
| * Papers that talk about adaptive, self managed software and run time configuration of adaptive software
* Papers talk about MDD approaches and in particular that focus on the synchronization between run time model and test model for adaptive software
* Papers talk about change propagation as a solution to the problem of runtime synchronization of tests for adaptive software
* Paper that talk about autonomic computing
* Papers include directly answer any one or more of our research questions;
* Paper talk about dynamic maintenance of adaptive software.
 | * Papers that are based on expert opinion
* Short papers, introduction to special issues, tutorials, and mini tracks
* Studies not related to research question
* Studies whose finding are unclear and ambiguous
* Preliminary conference versions of included journal papers
 |

The criteria covered three main issues pertaining to quality that need to be considered when appraising the studies identified in the review:

**Table 5 – Quality Assessment Criteria**

|  |  |
| --- | --- |
| S. No | Quality Assessment Criteria |
| 1 | Is the paper based on research (or is it merely a ‘‘lessons learned” report based on expert opinion)? |
| 2 | Is there a clear statement of the aims of the research? |
| 3 | Is there an adequate description of the context in which the research was carried out? |
| 4 | Was the research design appropriate to address the aims of the research? |
| 5 | Was the recruitment strategy appropriate to the aims of the research? |
| 6 | Was there a control group with which to compare treatments? |
| 7 | Was the data collected in a way that addressed the research issue? |
| 8 | Is there a clear statement of findings? |
| 9 | Is the study of value for research or practice? |

1. *Rigour*. Has a thorough and appropriate approach been applied to key research methods in the study?
2. *Credibility*. Are the findings well-presented and meaningful?
3. *Relevance*. How useful are the findings to the software industry and the research community?

The first three criteria represent the minimum quality threshold of the review and are used to exclude non-empirical research papers. Next five criteria are related to the rigor of the research methods employ to establish the validity of data collection tools and the analysis methods, and hence the trustworthiness of the findings. In addition, two criteria are related to the assessment of the credibility of the study methods for ensuring that the findings are valid and meaningful. More details can be find in Appendix B.

 Taken together, these criteria provide a measure of the extent to which we could be confident that a particular study’s findings could make a valuable contribution to the review. Each of the criteria will be graded on a dichotomous (‘‘yes” or ‘‘no”) scale. The quality assessment criteria are shown in Table 5. More details can be found in Appendix B

## Data Extraction

In the data extraction, data was extracted from each of the primary studies included in this systematic review according to a predefined extraction table. Table 6 enabled us to record full details of the articles under review and to be specific about how each of them addressed our research questions.

## References

1. Peyman Oreizy, Michael M. Gorlick, Richard N. Taylor, Dennis Heimbigner, Gregory Johnson, Nenad Medvidovic, Alex Quilici, David S. Rosenblum, and Alexander L Wolf, (1999), an architecture-based approaches to self-adaptive software, IEEE Intelligent Systems, vol. 14, no. 3, pp. 54-62.

**Table 6 – Data Extraction Form**

|  |  |
| --- | --- |
| Attribute | Description |
| Reference number | This indicates reference number of the paper considered for study |
| Authors | The authors of the paper |
| Year | The Year of the publication |
| Title | The title of the publication |
| Software | Does the paper apply to a particular type of software? If Yes: What type of the software? |
| Type of article  | Journal / conference / technical report  |
| Data collection  | How the data was collected e.g. interviews, questionnaires, measurement forms, observations, discussion, documents  |
| Concepts | The key concepts or major ideas in the primary studies  |
| Study aims  | The aims or goals of the primary study |
| System/tool | Does the paper specify any systems/tools to support adaptive software synchronization? If Yes: What are basic functions of the system/tool? |
| Evaluation technique | Does the paper include evaluation technique for adaptive software synchronization? If Yes: What is evaluation technique? |
| Study findings  | Major findings and conclusions from the primary studies.  |
| Relevance  | Relevance to the current research  |

1. Rahul Agarwal,(), Software Development vs. Software Maintenance.
2. Tariq M. King, Alain E. Ramirez, Rodolfo Cruz, and Peter J. Clarke (2007), An Integrated Self-Testing Framework for Autonomic Computing Systems
3. Hailpern, B.; Tarr, P.: Model-driven development: The good, the bad, and the ugly. In: IBM Systems Journal 45 (2006), No. 3, pp.451–461.
4. Laurence Tratt, (2008),A change propagating model transformation language, Bournemouth University, Poole, Dorset, BH12 5BB, United Kingdom
5. Jeffrey O. Kephart, David M. Chess, "The Vision of Autonomic Computing," Computer, pp. 41-50, January, 2003
6. IBM Autonomic Computing Architecture Team. An architectural blueprint for autonomic computing. Technical report, IBM, Hawthorne, NY, June 2006.
7. Zhang, J., Goldsby, H. J., and Cheng, B. H. 2009. Modular verification of dynamically adaptive systems. In *Proceedings of the 8th ACM international Conference on Aspect-Oriented Software Development* (Charlottesville, Virginia, USA, March 02 - 06, 2009). AOSD '09. ACM, New York, NY, 161-172.
8. Niebuhr, D. and Rausch, A. 2009. Guaranteeing correctness of component bindings in dynamic adaptive systems based on runtime testing. In *Proceedings of the 4th international Workshop on Services integration in Pervasive Environments* (London, United Kingdom, July 13 - 17, 2009). SIPE 09. ACM, New York, NY, 7-12.
9. Salehie, M. and Tahvildari, L. 2009. Self-adaptive software: Landscape and research challenges. *ACM Trans. Auton. Adapt. Syst.* 4, 2 (May. 2009), 1-42.
10. M. Aksit and Z. Choukair, “Dynamic, Adaptive and Reconfigurable Systems Overview and Prospective Vision,” Proc. 23rd Int’l Conf. Distributed Computing Systems (ICDCS ’03), pp. 84-89, May 2003.

**Appendix A – Detailed Search Strings**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| String # | High Level Search String | Detailed Search String | Review Quest. | Purpose |
| 1 | synchronization approaches for related models | *(Model OR software OR system OR development OR approach OR mechanism) AND (adaptive OR self adaptive OR adaptable OR adjustable OR autonomic) AND (configuration OR instrumentation OR installation) AND (changing OR modification OR revision OR alternation OR updating OR maintaining )* *AND (run-time OR run time OR dynamic OR operational-time OR running OR active-time*  *OR real-time) AND (synchronous OR concurrence OR simultaneous OR coincidence OR configuration OR traceable)*  | Q1 | To determine if there are synchronization approaches between related models in general and for adaptive software models with their test models in specific |
| 2 | Approaches that tackles the consistency and completeness during run time configuration for adaptive software | *(Model OR software OR system OR development OR approach OR mechanism) AND (adaptive OR self adaptive OR adaptable OR adjustable OR autonomic) AND (configuration OR instrumentation OR installation) AND (completeness OR entirety OR integrity OR perfection OR fullness OR wholeness) AND (consistency OR agreement OR harmony OR correspondence OR accordance) AND (changing OR modification OR revision OR alternation OR updating OR maintaining ) AND (propagation OR spreading OR distributing OR broadcasting) AND (run-time OR run time OR dynamic OR operational-time OR running OR active-time*  *OR real-time) AND (synchronous OR concurrence OR simultaneous OR coincidence OR configuration OR traceable)*  | Q2 | To find any approach reported in literature that use change propagation to improve consistency and completeness |
| 3 | Change propagation for synchronizing run time model of adaptive software’s  | *(Model OR software OR system OR development OR approach OR mechanism) AND (adaptive OR self adaptive OR adaptable OR adjustable OR autonomic) AND (configuration OR instrumentation OR installation) AND (changing OR modification OR revision OR alternation OR updating OR maintaining ) AND (propagation OR spreading OR distributing OR broadcasting) AND (run-time OR run time OR dynamic OR operational-time OR running OR active-time*  *OR real-time) AND (synchronous OR concurrence OR simultaneous OR coincidence OR configuration OR traceable)*  | Q2 | To determine if there are any change propagation technique used in the literature  |
| 4 | The limitations of the techniques in terms of the operational overhead of updating models | *(Model OR software OR system OR development OR approach OR mechanism) AND (adaptive OR self adaptive OR adaptable OR adjustable OR autonomic)* *AND (changing OR modification OR revision OR alternation OR updating OR maintaining )* *AND (run-time OR run time OR dynamic OR operational-time OR running OR active-time*  *OR real-time) AND (synchronous OR concurrence OR simultaneous OR coincidence OR configuration OR traceable) AND (limitation OR challenges OR threats OR dangers OR risks OR hazards )* | Q1 | To investigate the limitations of the approaches in terms of up-to-date models |
| 5 | Harnessing change propagation to synchronize the adaptive model of the software, with a runtime test model  | *(Model OR software OR system OR development OR approach OR mechanism) AND (adaptive OR self adaptive OR adaptable OR adjustable OR autonomic)* *AND (run-time OR run time OR dynamic OR operational-time OR running OR active-time*  *OR real-time) AND (synchronous OR concurrence OR simultaneous OR coincidence OR configuration OR traceable)* AND *(propagation OR spreading OR distributing OR broadcasting)*  | Q3 | To determine if the change propagation languages were developed can be applied to this problem |