Journal of
Information Technology & Software Engineering

Vieira, J Inform Tech Softw Eng 2012, 2:4
DOI: 10.4172/2165-7866.1000e112

Editorial Open Access

Challenges on Developing and Operating Trustworthy and Resilient Service Software Systems

Marco Vieira*

DEI/CISUC, University of Coimbra, Portugal

Introduction

Emerging large-scale, dynamic and open *heterogeneous service computing environments* applied to realize critical services, call for solutions that guarantee high trustworthiness and resilience. In practice, the current situation, as well as the expected evolution of complex service provisioning systems and infrastructures, shows a clear trend towards dominant characteristics such as extreme complexity, heterogeneity, mobility, scalability, dynamicity, and a very large scale of composable components and services.

To cope with the dynamic nature of large-scale systems and services, as well as with many other prominent practices, such as the use of agile software development methodologies where the requirements evolve during the system lifecycle, new methods and tools are needed. In practice, we need to researchnew methods, techniques and tools for improving the resilience and trustworthiness of future service software systems.

This paper describes the research approach being carried out at the Software and Systems Engineering Research Group of the Centre for Informatics and Systems of the University of Coimbra, to address the existing challenges on developing and operating trustworthy and resilient service software systems.

Challenges

The next paragraphs present the envisioned challenges. Achieving these requires the adaptation of the lifecycles and processes typically applied in the development of critical software services, as development takes place while the system (or part of it) is already under execution.

Support the development of resilient and trustworthy services and their composition

Developing resilient trustworthy services requires adequate development-time techniques and tools. In critical scenarios, we must reason about resilience in terms of system KPI–Key Performance Indicators (e.g. how can a design option prevent a fault from compromising a KPI?) and taking into consideration runtime dynamic adaptation [1]. A key goal is to create models, strategies and tools for incorporating KPI information and to combine runtime adaptation decisions in the development rationale. Other key aspects are: formal definition of the service behaviour, allowing the complete externalization of the service interface and resilience attributes; assessment and benchmarking of components and services, providing the means required for selecting the components and/or services that guarantee better levels of resilience; and resilient-specific development techniques that can be easily used by the developer and that can work in tandem with the runtime adaptation.

Allow continuous system Verification and Validation (V&V)

V&V has been successfully applied in traditional embedded critical systems as the main instrument for guaranteeing trustworthiness [2]. However, there are no adequate techniques for supporting V&V in the current and forthcoming large-scale and dynamic service software systems, as most of the existing solutions do not address aspects such as increased complexity, agility, evolution, and adaptation. This calls for techniques that support both development-time and runtime verification and validation, in particular: support for traceability of evolving requirements, coping with foremost features of agile development processes; explore regression in V&V to cope with the development style based on successive software releases; and support for runtime V&V of dynamic and evolving systems, allowing the online identification of changes, and the subsequent verification and validation of the updated system.

Support proactive system maintenance

To react quickly to changes (i.e., to be resilient), we need to fully exploit emerging monitoring techniques to support runtime anomaly detection and failure prediction algorithms, providing the support for preventive maintenance actions [3,4]. Supporting proactive maintenance calls for: an adaptive and scalable service-level monitoring infrastructure, which can be adapted according to changes that are either observed or predicted to occur; runtime stimulation techniques for externally stimulating the system for obtaining more data about the monitored infrastructure; anomaly detection and failure prediction techniques, with enhanced coverage that take advantage of the supplementary operational data about the infrastructure produced through adaptive monitoring and runtime stimulation; and maintenance strategies, selected, tuned and applied at runtime based on failure predictions and on their impact on the trustworthiness of the system.

Approach

The proposed approach is based on the integration of development-time and runtime techniques and processes that support the continuous development and maintenance of resilient and trustworthy service software systems. This calls for development techniques that take into account the potential dynamic adaptation of the system (and also take advantage of monitoring data from the runtime environment), verification and validation tools able to continuously portray the resilience of systems, and runtime maintenance techniques that allow proactively preventing the occurrence of failures. The following points introduce possible approaches for accomplishing the challengesdescribed above:

Development of resilient services

Incorporate KPIs information and runtime monitoring data in the development process: devise a methodology for software

***Corresponding author:** Marco Vieira, DEI/CISUC, University of Coimbra, Portugal, Tel: +351-2397-90068; Fax: +351-2397-01266; E-mail: mvieira@dei.uc.pt

Received September 23, 2012; **Accepted** September 24, 2012; **Published** September 26, 2012

Citation: Vieira M (2012) Challenges on Developing and Operating Trustworthy and Resilient Service Software Systems. J Inform Tech Softw Eng 2:e112. doi:10.4172/2165-7866.1000e112

development guided by Key Performance Indicators (KPIs), which must be present in every stage of the software life cycle, allowing to pinpoint the exact requirements, quality attributes, modules, code, V&V activities and technologies to each individual KPI. This way, if a KPI is not attained, the process to identify the source of the problem (when software related) would be fairly straightforward. Furthermore, we need to use runtime data to design development tools and techniques that prevent the occurrence of problems.

Support the formal definition of the service behavior in terms of resilience attributes: define a language that enables the description of a service from a resilience perspective. Mechanisms based on versioning allied with timeframes and time windows can be explored in to capture real-time behavior. Since the KPIs need to be made available and accessible to external parties which maybe located in any part of world, we may explore recent developments in the area of Semantic Web to enable access to KPIs data for reasoning and knowledge inference to better understand why certain runtime behaviors occur.

Support the assessment of software components and services: tools to assess the resilience of third-party components individually and as a whole, in order to verify which combinations would work better in practice. Furthermore, in many systems the tools and benchmarks being used need to adapt to changing conditions, since the experiments being done during development must comply with data collected at runtime. The goal is to use such data in a feedback loop to transparently update tools and workloads, used for assessment and benchmarking, on the development phase.

Development-time and Runtime Verification & Validation

Support traceability to evolving requirements and cope with successive software releases: provide tools that allow versioning requirements and tracking evolution. A potential approach is to use multidimensional traceability matrixes that describe the different versions of each requirement, allow mapping each version of each requirement with the system architecture and software code, and allow identifying interdependencies among existing requirements. The matrix and the existing metadata describing the pool of development time V&V checks and results from the previous system versions can be used to identify the checks that need to be repeated/updated and those that are still valid.

Support continuous assessment: Monitoring services and infrastructures are required for the runtime assessment of the system. This infrastructure that can be adapted according to changes, should rely on the idea of composing a dynamic network of monitors/checkers deployed concurrently with the creation/evolution of the composite network of services. The monitored information is the basis for runtime V&V activities during dynamic reconfiguration/composition [5]. A key issue is that the V&V techniques that can be applied at runtime may be limited.

Proactive maintenance

Runtime stimulation techniques: should be used for externally stimulating the real system or its virtual image for obtaining more data about the monitored infrastructure, in order to improve the accuracy and timing of existing approaches in detecting anomalies and predicting failures. Techniques on-line stressing, fault injection, forced exceptions, etc., should be studied to achieve this objective.A major challenge is the ability to encapsulate the stimulation in order to avoid uncontrolled propagation of anomalies not related to the normal operation of the infrastructure.

Anomaly detection and failure prediction techniques: should be based on supplementary operational data about the infrastructure produced through adaptive monitoring (as used for runtime V&V) and runtime stimulation. Theidea is to rely on the identification of variables and patterns of the infrastructure's operational profile by employing data mining and machine learning techniques, including variable (feature) selection methods and regression analysis. For failure prediction, instead of following the conventional techniques that just observe the operational state of the infrastructure, the proposed approach heavily relies on the additional data produced by the controlled runtime stimulation.

Maintenance strategies: should be selected, tuned and applied at runtime based on the results from monitoring and subsequent failure predictions and on their impact on the trustworthiness of the system [6].The maintenance actions should be determined based on one or more variables correlated to the degradation of the system state.

Conclusion

Nowadays, the composition of systems is shifting from development-time to runtime, resulting in dynamic service software systems that evolve after deployment in order to adapt to changes in the requirements and infrastructure. This way, we need to research disruptive techniques and tools for endowing resilience in complex, high-demand, large-scale, dynamically networked and critical service software systems, through a synergetic development, verification & validation, and maintenance, serving as key instrument for guaranteeing their trustworthiness and gaining confidence in their resilience.

In practice, we need methodologies, techniques, and tools that allow: 1) developing, deploying, and operating resilient service software systems, considering that they might evolve during development-time, and adapt during runtime; 2) verifying and validating service software systems at both development-time and runtime, in a way that allow evaluating and assuring their trustworthiness and resilience; and 3) improving system resilience by performing pro-active maintenance, preventing the occurrence of failures.

References

1. Cai J, Liu X, Xiao Z, Liu J (2009) Improving supply chain performance management: A systematic approach to analyzing iterative KPI accomplishment. Decision Support Systems 46: 512-521.
2. Tran E (1999) Verification/Validation/Certification. Carnegie Mellon University.
3. Yam RCM, Tse PW, Li L, Tu P (2001) Intelligent predictive decision support system for condition-based maintenance. Int J Adv Manuf Technol 17: 383–391.
4. Salfner F, Lenk M, Malek M (2010) A Survey of Online Failure Prediction Methods. ACM Computing Surveys 42: 1-68.
5. Sokolsky O, Havelund K, Lee I (2011) Introduction to the special section on runtime verification. Int J Softw Tools Technol Transfer 14: 243-247.
6. Bruns P (2002) Optimal maintenance strategies for systems with partial repair options and without assuming bounded costs. European Journal of Operational Research 139: 146–165.