Enforcing Safety Requirements for Industrial Automation Systems at Runtime

Position Paper

Wikan Sunindyo, Martin Melik-Merkumians, Thomas Moser and Stefan Biffl
Christian Doppler Laboratory "Software Engineering Integration for Flexible Automation Systems"
Vienna University of Technology
Vienna, Austria
{*firstname.lastname*}@tuwien.ac.at

***Abstract*—Current industrial automation systems are becoming more and more complex, and typically involve different phases of engineering, such as design time and runtime. System requirements, which are usually elicited during design time by engineers, currently are not sufficiently represented at runtime, like the runtime enforcement of safety requirements for industrial automation systems. Such kind of enforcement usually is very hard to model and predict at design time. Hence, the need exists to capture and manage safety requirements at design time and runtime, since safety requirements of industrial automation systems may lead to high risks if not addressed properly. In this position paper, we introduce a safety requirements enforcement framework and the using of Boilerplates for requirements elicitation and by explicitly modeling the runtime requirements knowledge for further application. We illustrate and evaluate the approach with data from a real-world case study in the area of industrial process systems. Major result was that the Boilerplates and explicit engineering knowledge are well suited to capture and enforce runtime safety requirements of industrial automation systems.**

Keywords: industrial automation systems, safety requirements, requirements at runtime, requirements elicitation.

I. INTRODUCTION

Complex industrial automation systems, such as manufacturing plants, power plants or process plants, typically involve heterogeneous components in different engineering phases, e.g., design time and runtime. In the design time phase, engineers can elicit requirements of system users and other system stakeholders easily by using different methods, such as interviews or questionnaires. These design-time requirements are then documented in the software requirement specification for further purposes, e.g., for design, implementation, and testing of the final product [10]. However in complex industrial automation systems, not all requirements can easily be modeled and predicted at design time, e.g., safety requirements.

Some safety requirements can be derived from the initial needs of the users or other stakeholders, but some other safety requirements can be derived only during or even after the systems runtime. Hence, there exists the need to capture and manage safety requirements both at design time and at runtime, since not properly addressed safety requirements of industrial automation systems may lead to high risks and even severe damages to people and environment.

In this position paper, we introduce a safety requirements enforcement framework and propose the using Boilerplates[1] for requirements elicitation and explicitly modeling the runtime requirements knowledge for further application using ontologies for knowledge representation. We use an educational model of an industrial process plant as a use case for industrial automation systems. The educational model is used for simulation of complex industrial batch process, and consists of different components, such as tanks, valves, pipes and sensors, that are connected to each other and are controlled using industrial control software. The usage of the educational model of an industrial process plant for enforcing safety requirements at runtime can be used to illustrate the actual situation of real industrial automation systems, typically involving much more complex and interconnected components. In this work, the elicitation of safety requirements is performing by analyzing several scenarios in which high-risk accidents could happen to the system. By observing these scenarios, we collect the safety requirements and model them using Boilerplate notation. These safety requirements later are used as the basis for linking to design time information in order to validate constraints or rules at runtime.

Initial results show that the analysis of different scenarios of possible failures of the industrial process plant model at runtime allows for a more efficient and effective elicitation and management of runtime safety requirements, which are not easily predicted or modeled at design time. These safety requirements are very useful for avoiding the system from fatal accidents that could destroy the machinery and hamper the production processes.

The remainder of this position paper is structured as follows: Section 2 shortly summarizes relevant related work, while Section 3 describes the research goal. Section 4 explains the solution approach and details on the use

[1] http://www.requirementsengineering.info/boilerplates.htm

case using the educational model of an industrial process plant. Section 5 discusses the initial findings, summarizes benefits and limitations of the proposed approach the results, concludes the paper and identifies further work in the research area.

II. Related Work

This section summarizes related work on industrial automations systems and requirements elicitation approaches at runtime.

A. Industrial Automation Systems

Complex industrial automation systems need to be flexible to adapt to changing business requirements and to become more robust against failures during runtime.

Melik-Merkumians *et al.* proposed an ontology-based Engineering Knowledge Base [7] to provide relevant design time and runtime engineering knowledge in machine-understandable form to be able to better identify and respond to failures. The authors used an example of an educational model of an industrial process plant to simulate complex industrial batch processes, such as the processes typically implemented in like refineries, breweries, or pharmaceutical plants. It consists of two tanks holding the process fluids. The liquid levels of the tanks are checked by several analog and digital sensors. The lower tank also contains a pump which either transports the process fluid into the upper tank, or is used to mix up the process fluid in the lower tank. Also the lower tank contains a heater and a temperature sensor to heat the process fluid to specified temperatures. This educational model of an industrial process plant is also used as use case in this paper to demonstrate the proposed requirements elicitation at enforcement of safety requirements at runtime for industrial automation systems.

B. Eliciting Requirements at Runtime

Requirements elicitation at runtime is a new research field which contains a lot of potential open issues to explore further.

Some approaches have already been proposed in order to elicit safety-critical requirements of software-intensive systems at design time. For example, Cruickshank *et al.* [2] proposed a validation metrics framework for safety-critical software intensive systems. By using Goal Question Metrics (GQM) and Goal Structuring Notation (GSN) approaches, they build their framework and apply it to a safety-critical, software-intensive surface-to-air missile system. The application area is by now limited, but it is possible to generalize it to be used in other specific areas. However, as the requirements elicitation and validation by now is primarily done at design time, there are still many requirements can only be detected in the runtime. Hence we need to generalize the requirements elicitation approach to be used also during runtime.

By now, the term "requirements at runtime" is not clearly defined and still needs further research. Sawyer *et al.* [9] discussed the definition of requirements at run time and defined it as more than just a representation of a behavioral specification. One needs to be aware of the current state of all affected models and their entities, goals, adaptations, tracing links, entities, and preferences, in order to successfully manage requirements at runtime.

We use information available only at runtime to elicit requirements that cannot be detected at design time. This information could be stored in explicit and machine-understandable form for further purposes, such as linking to design time information. Jian *et al.* [6] proposed a goal-oriented requirements modeling for running systems and introduce the usage of such explicit knowledge bases to store the requirements collected at runtime. However, the connections to design time need further research.

Farfeleder *et al.* [4] present an approach for representing the requirements of the systems. In this paper, they propose to use formal languages for requirements elicitation as an alternative to the natural language. They introduce tool-support that semi-automatically transforms natural language requirements into semi-formal Boilerplate requirements. The term Boilerplate was first used by Hull *et al.* [5] to refer to a textual requirement template. A Boilerplate consists of a sequence of attributes and fixed syntax elements (FSEs). For example, the Boilerplate "<*system*> *shall* <*action*>". <*system*> and <*action*> are attributes, while *shall* is an FSE in this Boilerplate. We combine several Boilerplates by means of simple concatenation to keep the number of required different Boilerplates low, while at the same time providing high flexibility.

III. Research Goal

This section describes the research goal and main research issue we want to discuss in order to enforce the safety requirements at runtime.

Safety requirements usually are derived at design time using a set of different scenarios [1]. Daramola *et al.* [3] proposed a conceptual framework to analyze the safety requirements semantically, by using Hazard and Operability (HAZOP) and Failure Mode and Effect Analysis (FMEA) analysis approaches. These methods can be used to elicit and analyze the safety requirements of systems at design time. However, the need to enforce that the safety requirement items are really fulfilled during runtime is still a big open question.

We formulate our research goal as *how to establish a set of steps and rules to enforce and ensure that the safety requirement items we have derived in the design time are really followed and fulfilled at system runtime*. We use the educational model of an industrial process plant as our use case; this use case can easily be generalized for larger industrial automation systems.

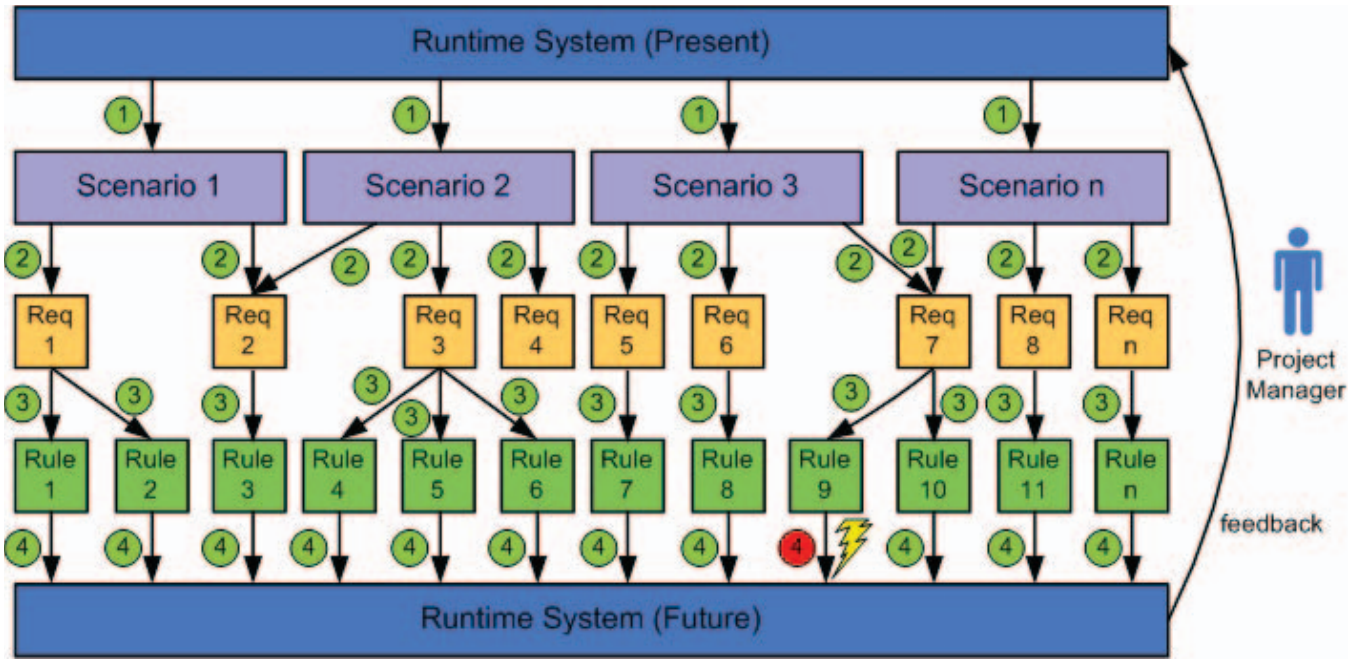

Figure 1. Framework of Safety Requirements Enforcement

IV. Solution Approach

In this section, we present the description of the proposed solution approach to address the overall research goal introduced in the previous section. The solution approach is presented using the educational model of an industrial process plant, which consists of heterogeneous components that are connected to each other. From this so-called "tank model", we derive safety requirement that are required to prevent the tank model from damage

A. General Description

The understanding of the general description of the tank model is essential for an easier understanding of the safety requirements we describe later. The illustration of the tank model as design time model and as an image of the real running system is shown in Figure 2 (design time on the left hand side, runtime on the right hand side). For the design time model, we use the P&ID (Piping and Instrumentation Diagram) of the educational model of an industrial process plant located in the Odo-Struger-Laboratory[2].

As described in detail in [7] and [8], the tank model consists of two tanks, *T101* and *T102*, which are situated on different height levels. The lower tank (*T101*) contains three level sensors (a critical upper level (*B114*), a lower level (*B104*), and a critical lower level lever (*B113*)), a heater (*E104*), and a temperature sensor (*S111*). The upper tank (*T102*) contains an ultrasonic level sensor (B101), which measures the distance of the liquid surface to the upper tank closure, and a float lever (*S112*) which represents the critical low level of the liquid. The two tanks are connected by a set of pipes which are opened or closed using a set of valves. The liquid transportation from the lower to the upper tank is performed using a pump (*P101*). The actual liquid flow caused by the pump is measured by a flow meter (*B102*).

This process configuration in this educational model of an industrial process plant represents a typical plant configuration in the process industry and thus can be used to simulate the components and their behavior.

B. Safety RequirementsEnforcement at Runtime

The framework for safety requirements enforcement at runtime is illustrated in Figure 1. In the following paragraphs, the steps for safety requirements elicitation and their enforcement at runtime are explained.

1) Define runtime critical scenarios. The runtime safety requirements elicitation process cannot be done by assuming normal conditions. Thus, we define critical scenarios based on possible dangerous situations or risks that could happen to the tank model and disturb or even damage the system and its components. We obtain the information about such scenarios from domain experts using methods like interviews or questionnaires.

2) Model the requirements of critical scenarios using Boilerplate notation. We now formulate the safety requirements we have obtained from Step 1 using Boilerplate notation. Boilerplate notation is a requirement elicitation notation, which consists of templates and their attributes filled with concrete attribute values. By using Boilerplate notation, it is easier for engineers to define the

[2] The industrial automation systems laboratory of the Automation and Control Institute (ACIN), Vienna University of Technology (http://www.acin.tuwien.ac.at).

system and its abilities as requirement items, also including safety requirements, since only missing values have to be filled out in the available templates. Furthermore, the usage of exact Boilerplate templates guarantees a well-defined and at least semi-formal requirements definition, which later can be processed automatically.

3) Transform each requirement into if-then-else rules. After defining safety requirement items in the previous step, we transform each safety requirement into a set of "if-then-else" rules. The usage of "if-then-else" rules allows for an easier rule evaluation at runtime. We easily can evaluate whether all tank model components fulfill all suitable rules without any violation.

4) Evaluate each rule periodically. The tank model components later are checked periodically regarding all defined "if-then-else" rules to enforce that all safety requirements are really fulfilled by the running system.

C. *Exemplary Runtime Scenarios*

We now illustrate the proposed method using two critical runtime scenarios (RTS). These scenarios were originally described in [7] and [8], in this paper we used them as example for real-life usage scenarios of the proposed transformation approach for the transformation of design time safety requirements to rules which can be used to enforce these safety requirements at runtime.

RTS-1: Undetected valve failure in the pump pipe section. In this scenario, the liquid in tank *T101* is pumped into tank *T102*. Therefore the valve *V104* must be closed and the valves *V101* and *V109* must be opened. We assume that one or both valves that should be opened are defective and therefore are unable to get into the required position. Such a situation can lead to damage of the pump, as it is either pumping against the impregnable resistance of valve *V101*, or it will be soon out of liquid as the valve *V109* does not provide further liquid supply, which may lead to damage of the pump, e.g., in case of rotary pumps.

To avoid such equipment failures, it is common to use a flow meter *B102* which can be used to compare the actual flow with the anticipated flow value. If these two values differ too much, the pump can be put into an emergency shutdown mode, in order to prevent any further damage of the used equipment. We now define the safety requirements for this scenario by using the Boilerplates shown in Listing 1. Figure 2 illustrates the requirements elicitation processes and relationships between the design time configuration and runtime safety requirements in Boilerplate.

Boilerplate Templates

- `BP2: The <system> shall be able to <action> <entity>.`
- `BP62: The <system> shall allow <entity> to be <state>.`

Filled Boilerplate Templates

- `The pump P101 shall be able to pump liquid from tank T101 to tank T102.`
- `The tank model shall allow valve V104 to be closed.`
- `The tank model shall allow valve V101 to be opened.`
- `The tank model shall allow valve V109 to be opened.`
- `The flow meter B102 shall be able to compare the actual flow with the anticipated flow value.`

Listing 1. Boilerplates for RTS-1.

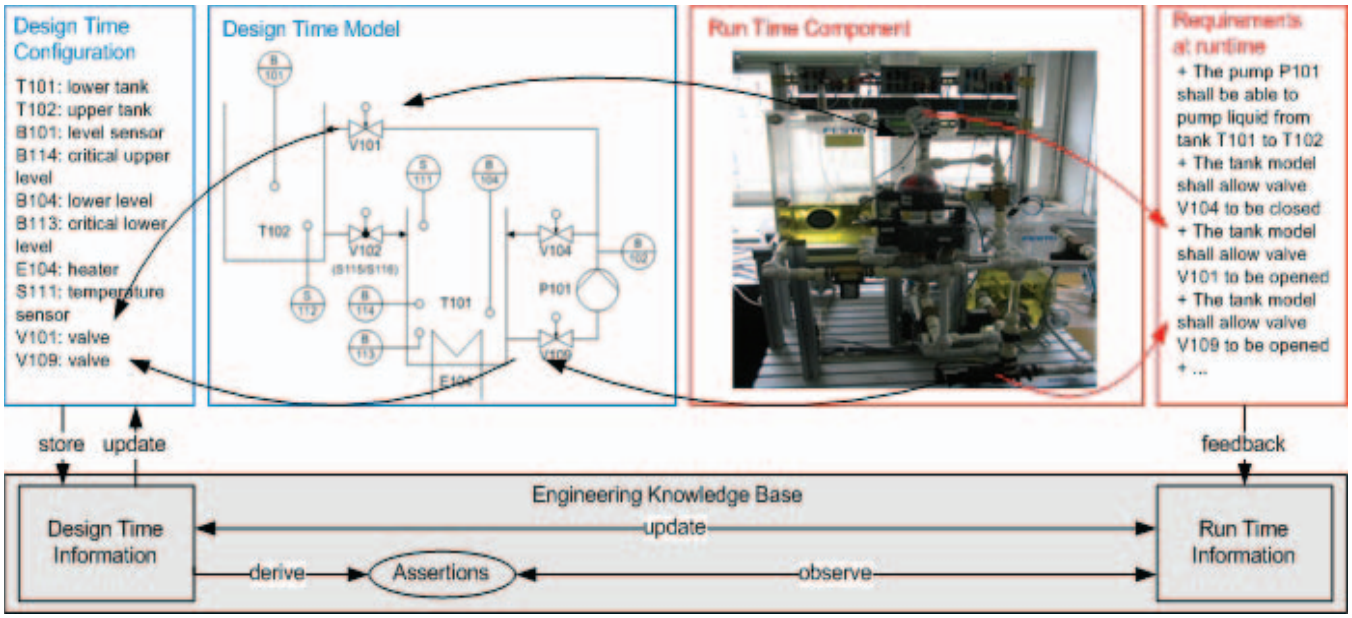

Figure 2. Safety Requirements Elicitation at Runtime in the Educational Process Plant (adapted from [7]).

After that, we transform the requirements into if-then-else rules as shown in Listing 2. The rules are checked with all required and suitable components (the pump, the tanks, the valves, the sensors and the flow meters) to enforce that the elicited safety requirements are really fulfilled at system runtime.

```
while (P101.isActive ())
  for (B102.getSensorEvent() as x)
    for (B102.getSensorEvent() as y)
      if (x.getTimestamp () < y.getTimestamp())
        if (NOT (y.getLevel() > x.getLevel ()))
          <<raise alarm>>
```

Listing 2. If-Then-Else Rules for RTS-1 [7].

RTS-2: Liquid level of heated tank. When the heater *E104* is turned on, it is required that there is any liquid in the tank *T101* to prevent the heater from overheating and therefore from any other damage caused by this overheating. The level of liquid should always be between the critical lower level (*B113*) and critical upper level (*B114*). We define the requirements in the Boilerplate notation as shown in Listing 3.

Boilerplate Templates

- BP2: The <system> shall be able to <action> <entity>
- BP36: ...while <operational condition>
- BP63: If <operational condition>
- BP64: ...the <system> shall <action>

Filled Boilerplate Templates

- The heater E104 shall be able to heat the liquid while there is liquid in tank T101.
- The sensor B113 shall be able to detect the level of liquid.
- The sensor B114 shall be able to detect the level of liquid.
- The sensor B104 shall be able to detect the level of liquid.
- If there is no liquid in tank T101, the heater E104 shall automatically shut down.

Listing 3. Boilerplates for RTS-2.

For this runtime scenario, a larger number of Boilerplates is used. We then transform the requirements into if-then-else rules as shown in Listing 4.

```
while (E104.isActive ())
  for (B113.getSensorEvent() as x)
    for (B114.getSensorEvent() as y)
      for (B104.getSensorEvent() as z)
        if ( (z.getLevel() ≤ x.getLevel () or
              (z.getLevel() ≥ y.getLevel())
             E104.shutDown()
```

Listing 4. If-Then-Else Rules for RTS-2.

V. Discussion and Conclusion

The requirements of complex industrial automation systems, e.g., safety requirements, cannot easily be modeled and predicted at design time. Some safety requirements can be derived from the initial needs of the users or other stakeholders, but some other safety requirements can be derived only during or even after the systems runtime. Hence, there exists the need to capture and manage safety requirements both at design time and at runtime, since not properly addressed safety requirements of industrial automation systems may lead to high risks and even severe damages to people and environment.

In this position paper, we proposed a method to enforce safety requirements at runtime by using Boilerplates for requirements elicitation and by explicitly modeling the runtime requirements knowledge for further application using ontologies for knowledge representation. We initially evaluate the proposed approach using an educational model of an industrial process plant for enforcing safety requirements at runtime to illustrate the actual situation of real industrial automation systems.

Initial results show that the analysis of different scenarios of possible failures of the industrial process plant model at runtime allows for a more efficient and effective elicitation and management of runtime safety requirements, which are not easily predicted or modeled at design time. Perceived benefits of the proposed approach are the possibility of using the approach to check whether safety requirements are really applied and fulfilled during the system runtime. This safety requirement enforcement helps avoiding severe damages of the industrial automation system and its components. In contrast, we know that the initial results are somehow limited, since there is further research needed to allow for the implementation and statistical analysis of empirical studies. Furthermore, the scenarios discussed in this paper are limited and therefore we need to described and classify a larger number of scenarios to provide real-world usability of the proposed approach.

The major lessons learned during the initial research work are on the one hand side that the usage of a quite common method for requirements elicitation, such as the Boilerplate notation, generally enables an easier safety requirements elicitation at design time. On the other hand side, the transformation from Boilerplate notation into if-then-else rules is intuitively understandable and can be used to check whether a system is really following a set of rules using periodical rule checking mechanisms.

Further work will include an improvement of the Boilerplate notation for requirements elicitation, as well as the evaluation of the usage of other possible notations for the formalization of natural language requirements. We will define and use more example scenarios to derive additional safety requirements, which we then plan to classify into classes of scenarios and types of safety requirements to make the safety requirements easier to handle. We also plan to optimize the if-then-else rules and add reasoning support for evaluation and checking more complex runtime processes based on the explicitly stored design time knowledge. Finally, we will perform comprehensive empirical studies to measure the improvement provided by the proposed approach.

ACKNOWLEDGMENTS

This work has been supported by the Christian Doppler Forschungsgesellschaft and the BMWFJ, Austria. The authors also want to acknowledge the inputs from and discussions with Stefan Farfeleder, Inah Omoronyia and Olawande Daramola. Furthermore, the authors want to thank the domain experts at their industry partners for their valuable inputs.

REFERENCES

[1] K. Allenby, and T. Kelly, "Deriving safety requirements using scenarios," in Fifth IEEE International Symposium on Requirements Engineering (RE 2001), 2001, pp. 228-235.

[2] K. J. Cruickshank, J. B. Michael, and S. Man-Tak, "A Validation Metrics Framework for safety-critical software-intensive Systems," in IEEE International Conference on System of Systems Engineering (SoSE 2009) 2009, pp. 1-8.

[3] O. Daramola, T. Stålhane, T. Moser, and S. Biffl, "A Conceptual Framework for Semantic Case-based Safety Analysis," in 16th IEEE International Conference on Emerging Technologies and Factory Automation, Toulouse, France, 2011, p. accepted for publication.

[4] S. Farfeleder, T. Moser, A. Krall, T. Stalhane, H. Zojer, and C. Panis, "DODT: Increasing requirements formalism using domain ontologies for improved embedded systems development," in IEEE 14th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS 2011), 2011, pp. 271-274.

[5] E. Hull, K. Jackson, and J. Dick, *Requirements Engineering*: Springer, 2005.

[6] Y. Jian, T. Li, L. Liu, and E. Yu, "Goal-oriented requirements modelling for running systems," in First International Workshop on Requirements@Run.Time (RE@RunTime 2010), 2010, pp. 1-8.

[7] M. Melik-Merkumians, T. Moser, A. Schatten, A. Zoitl, and S. Biffl, "Knowledge-based Runtime Failure Detection for Industrial Automation Systems," in 5th Workshop on Models@run.time at the ACM/IEEE 13th International Conference on Model Driven Engineering Languages and Systems (MODELS 2010), Oslo, Norway, 2010, pp. 108-199.

[8] M. Melik-Merkumians, A. Zoitl, and T. Moser, "Ontology-based fault diagnosis for industrial control applications," in Emerging Technologies and Factory Automation (ETFA), 2010 IEEE Conference on, 2010, pp. 1-4.

[9] P. Sawyer, N. Bencomo, J. Whittle, D. M. Berry, and A. Finkelstein, "Foreword: First Workshop requirements@run.time," in First International Workshop on Requirements@Run.Time (RE@RunTime 2010), 2010, pp. 1-2.

[10] I. Sommerville, *Software Engineering*, 8th ed., p. 840, Essex, England: Pearson Education Limited, 2007.