LETTER

A Framework to Integrate Public Information into Runtime Safety Analysis for Critical Systems

Guoqi LI[†a)], ***Member***

SUMMARY The large and complicated safety-critical systems today need to keep changing to accommodate ever-changing objectives and environments. Accordingly, runtime analysis for safe reconfiguration or evaluation is currently a hot topic in the field, whereas information acquisition of external environment is crucial for runtime safety analysis. With the rapid development of web services, mobile networks and ubiquitous computing, abundant realtime information of environment is available on the Internet. To integrate these public information into runtime safety analysis of critical systems, this paper brings forward a framework, which could be implemented with open source and cross platform modules and encouragingly, applicable to various safety-critical systems.

key words: *safety critical systems, runtime analysis, software framework, safe reconfiguration*

1. Introduction

Safety-critical systems are those systems whose failure could result in loss of life, significant property damage, or damage to the environment. Now, safety-critical systems are becoming more and more common and powerful, for instance, the autonomous cars, home medical devices and civilian UAV (Unmanned Aerial Vehicles) are hopeful to enter our daily life in recent future. These systems are large and complicated and need to keep changing to accommodate ever-changing objectives and environments. The concept of "Open Systems Dependability" [1] was put forward to indicate the new insights and challenges on dependability of ever-changing systems. Runtime reconfigurability, which is of great importance for open systems dependability, could allow the systems be adapted every possible scenario, even if it was not foreseen at design time [2]. Accordingly, runtime safety analysis is proposed to direct runtime reconfigurations [3], [4].

Information acquisition of external environment is crucial for runtime safety analysis of critical systems. Traditionally, this information is obtained from sensors and detectors [3]. However, with the rapid development and popularization of web services, mobile internet and ubiquitous computing, abundant realtime information of environment, such as weather and road conditions, topography changes, infrequent events etc., is accessible easily and freely from the Internet. A former research has applied such information into realtime analysis of personal safety by providing smartphone applications [5].

As a further matter, the public information obtained from the Internet should also be valuable for runtime safety analysis of critical systems. Compared with the private information of critical systems, such as data from sensors and detectors, the public information has broader vision. Consider a potential application scenario: a low-altitude unmanned helicopter usually measures the wind speed and direction with sensors, but the detection range is limited to its current position, so the UAV could not predict the dangerous chaotic airflows between two big buildings. However, if it could get the near topographic information from Google Earth and intercity wind speed and direction from corresponding web services, the dangerous chaotic airflows would then be predicted and dodged by reconfiguration of flight control or replanning of flight route.

To integrate the public information from the Internet into runtime safety analysis of critical systems, a framework and the implementation of its key modules are provided in the following section.

2. Framework and Implementation of Key Modules

Figure 1 briefly illustrates the composition of the framework. The rectangle in the center of the figure represent three modules. "JavaScript objects" are responsible for information acquisition from the Internet and "Native objects"

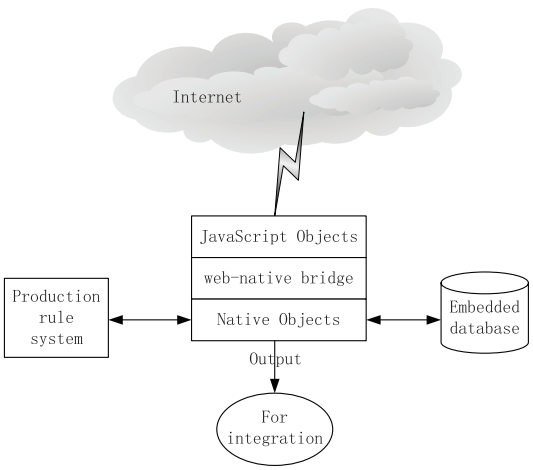

Fig. 1 The brief diagram of the framework to integrate public information into runtime safety analysis for critical systems.

Manuscript received September 17, 2013.
Manuscript revised November 8, 2013.

[†]The author is with Science and Technology on Reliability and Environmental Engineering Laboratory, School of Reliability and System Engineering, Beihang University (BUAA), Beijing China.
a) E-mail: gqli@buaa.edu.cn

DOI: 10.1587/transinf.E97.D.981

Copyright © 2014 The Institute of Electronics, Information and Communication Engineers

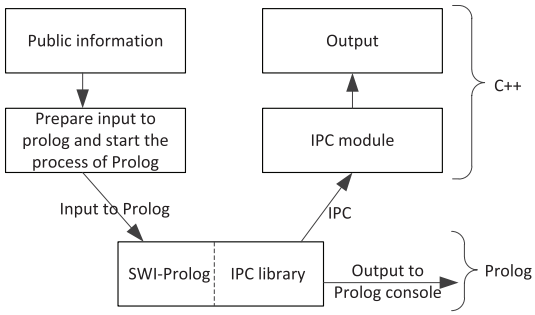

Fig. 2 The schematic diagram of integration SWI-Prolog with C++.

are able to access and manipulate local resources. These two kinds of objects communicate mutually through the "web-native bridge". Additionally, an "Embedded database" is needed for permanent and transient data storage. Finally, the analysis results are output "For integration".

As mentioned in the former section, realtime information of environment could be obtained from the Internet. For instance, road condition and topography data could be obtained from Google Maps and Goolge Earth respectively, both of which provide JavaScript API (Application Program Interface) for accessing relevant data conveniently. In the field of web programming, both at present and in the foreseeable future, JavaScript is in a dominant position, and network services usually provide interface in the form of Javascript API. Consequently, by supporting JavaScript, the framework is definitely qualified with the ability of information acquisition from the Internet.

The role of the "Production rule system", shown in the left side of the figure, is to process information obtained from the Internet. The raw data are usually of large quantities and should be filtered and recognized as specified, namely, meaningful information for safety. The logic of the process is coincident with the production rule system. We resort to SWI-Prolog as a rule engine and implement the production rule system with forward chaining, which is one of the two main methods of reasoning supported by Prolog. The substantial safety analysis could be written in forms of rules and conducted by the production rule system.

Additionally, Fig. 2 shows the method of integrating SWI-Prolog with C++ by an IPC (Inter Process Communication) based method. Prolog is a powerful logic programming language and SWI-Prolog provides independent consoles, multi-thread mechanisms, build-in predicates and etc. When calling a Prolog program from C++ through the interface of SWI-Prolog, additional threads or processes will be created and the results of the Prolog program will be output to the console of SWI-Prolog, instead of returning to C++, so the IPC library is necessary [6] to communicate with C++ mutually at realtime. In the figure, the components written in C++ belong to the module of "Native objects".

Last but not least, "web-native bridge", shown in the center of the Fig. 1, is another key module, and logically, the crucial part for the framework. It connects the JavaScript objects and native objects, so that both events occurred in the Internet and in the system itself could trigger reactions in the framework.

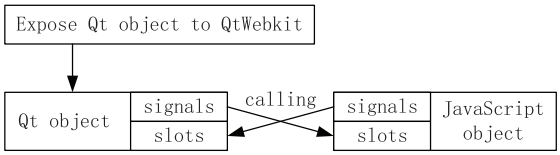

Fig. 3 The steps of connecting Qt object and JavaScritp object with QtWebKit. Signals and slots are used for communication between objects.

QtWebkit is the best candidate for implementing the web-native bridge. Of course QtWebKit is not the only bridge technology out there. NPAPI (Netscape Plugin Application Programming Interface), for example, is a long-time standard or web-native bridging. Due to Qt's meta-object system, full applications built partially with web-technologies are much easier to develop. NPAPI, however, is more geared towards cross-browser plugins, due to it being an accepted standard [7].

Three steps are needed to connect JavaScript objects and Qt objects with QtWebKit: first of all, expose Qt object to QtWebKit and then connect the singles and slots of Qt objects and JavaScript objects respectively according to Fig. 3. Here, signals and slots are used for communication between objects, which is an alternative to the callback technique. A signal is emitted when a particular event occurs and a slot is a function that is called in response to a particular signal. Briefly, slots are a kind of specific member function of an object, which could be called by other objects through emitting corresponding signals, even though the slots and singles are implemented with different languages.

Until now, there is no any difficulties for ordinary engineers to implement the framework. Additionally, the well-commented source codes of the components illustrated in Figs. 2 and 3 are available by email gqli@buaa.edu.cn or keep_thinking@hotmail.com.

3. Conclusions and Future Works

The main contribution of this paper is to provide a new idea to integrate the public information from the Internet into the runtime safety analysis of critical systems. A framework, which could be implemented with open source and cross platform modules, is presented to prove the feasibility of the idea.

In the future, we plan to construct a concrete implementation of the framework for an open source low-altitude UAV, and the evaluation of the application will be given in details.

Any discussion about the topic is welcome by emails mentioned above. This work is supported by the Fundamental Research Funds for the Central Universities (Program No.YWF-13-B05-001).

References

[1] M. Tokoro, "White paper of deos project version 3.0a," Tech. Rep.,

Japan Science and Technology Agency, Dec. 2011.
[2] L. Sterpone, M. Porrmann, and J. Hagemeyer, "A novel fault tolerant and runtime reconfigurable platform for satellite payload processing," IEEE Trans. Comput., vol.62, no.8, pp.1508–1525, Aug. 2013.
[3] K. Ostberg and M. Bengtsson, "Run time safety analysis for automotive systems in an open and adaptive environment," Proc. Workshop ASCoMS (Architecting Safety in Collaborative Mobile Systems) of the 32nd International Conference on Computer Safety, Reliability and Security (SAFECOMP), 2013.
[4] C. Priesterjahn, C. Heinzemann, W. Schafer, and M. Tichy, "Runtime safety analysis for safe reconfiguration," Proc. 10th IEEE International Conference on Industrial Informatics (INDIN), 2012.
[5] J. Ballesteros, B. Carbunar, M. Rahman, N. Rishe, and S. Iyengar, "Towards safe cities: A mobile and social networking approach," IEEE Trans. Parallel Distrib. Syst., vol.99, no.8, pp.1–14, Aug. 2013.
[6] J. Rosenwald, Transparent Inter-Process Communications (TIPC) libraries, 6.5.2 ed. Available at http://www.swi-prolog.org/pldoc/package/tipc.html
[7] The QtWebKit Bridge, qt5.0 ed., 2013. Available at: http://qt-project.org/doc/qt-5.0/qtwebkit/qtwebkit-bridge.htm