Journal of Network and Computer Applications 34 (2011) 1441–1446

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Email FI identification and resolution with model checking

André Vicente Calvinho [a,1], Rui Gustavo Crespo [b,*]

[a] Military Academy, Rua Gomes Freire, 1169-203 Lisboa, Portugal
[b] INESC-ID/Technical University of Lisbon, DEEC Av. Rovisco Pais, 1049-001 Lisboa, Portugal

ARTICLE INFO

Article history:
Received 12 July 2010
Received in revised form
12 February 2011
Accepted 10 March 2011
Available online 16 March 2011

Keywords:
Email features
Feature interaction
Feature resolution
Feature interdiction
Model checking

ABSTRACT

Internet applications, such as Email, VoIP and WWW, have been enhanced with features. However, the introduction and modification of features may result in undesired behaviors, and this effect is known as feature interaction ("FI"). Among other methods, constraint logic programming and model checking have been adopted to address the two main problems in telephony FIs: detection and resolution.

In this paper, we show that model checking is also suitable to detect FIs in more complex domains and we use Email features as an example. Moreover, FI detection may be simultaneously analyzed by model checking tools.

Finally, we analyze the implementation performance against a number of parties and message types using the CPAchecker tool. Model checking reveals a superior performance against constraint programming, for the case of FI detection with FI occurrence.

© 2011 Elsevier Ltd. All rights reserved.

1. Introduction

Nowadays, many Internet applications have been enhanced with several features. The problem occurs when some of these features, working fine alone, interact. That problem is known as feature interaction, or FI for short. FIs have been observed in several Internet applications, such as Electronic mail ("Email") (Hall, 2000), World Wide Web ("WWW") (Weiss, 2003) and Voice over IP ("VoIP") (Lennox and Schulzrinne, 2000).

Examples 1.1, 3.3 and 3.4 depict three Email feature interactions.

Example 1.1. Suppose that Alice subscribes to the feature ForwardMessage, as well as Bob, and each feature is configured to forward messages to each other. When a party receives a message, that message will bounce between the two parties forming a loop cycle. This is an undesirable behavior.

Despite the unexpected results, users must decide if FIs represent undesirable interactions, or not. For example, the WWW Refresh feature forms a loop to itself. However, if the web page depicts a clock, Refresh becomes an acceptable interaction.

The increasing number of FIs, and the inconvenience they are causing, led industry and researchers to meet regularly at the Feature Interactions in Telecommunications and Software Systems conferences, 10 of which have been held from 1992 to 2009.

Three basic problems have been studied (Bouma and Velthuijsen, 1994): *avoidance*, *detection* and *resolution*. Avoidance means to intervene at the protocol or design stages to prevent FIs, before features are executed. Due to the distributive characteristic of Internet, where every node is unaware of features subscribed by other users, avoidance is not considered here. Detection aims at the identification of FIs, with suitable methods. In the resolution, actions are exercised runtime over triggered features, which averts FIs.

In this paper we focus on Email FI detection and resolution, with particular interest on the 10 most widely known features (Hall, 2000).

1.1. FI detection approaches

Calder et al. (2003) showed how programming languages, such as Promela (Calder and Miller, 2001), CSP (Hoare, 1978) and LOTOS (Gorse et al., 2006), may be used to specify features. Methods explored so far in FI detection include simulation (Thomas, 1997), model checking (Plath and Ryan, 2000; Calder and Miller, 2001), theorem proving (Gammelgard and Kristensen, 1994) and prediction (Crespo, 2008).

In this paper we adopt model checking approach. Model checking deals with the verification of temporal properties over program models (Baier and Katoen, 2008). Model checking tools generate an output that indicates if the program is safe or not. If the program is unsafe, the outcome is complemented with a counter-example.

To specify feature specifications, we adopted C language (Kernighan and Ritchie, 1978), because it is the choice language

* Corresponding author. Tel.: +351 21 8417626; fax: +351 21 8417 499.
E-mail addresses: andrecalvinho@gmail.com (A.V. Calvinho), R.G.Crespo@comp.ist.utl.pt (R.G. Crespo).
[1] Fax: +351 213186988.

1084-8045/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jnca.2011.03.027