Extracting Behavioral Models From Service Implementations

Ioana Şora [1,2] and **Doru-Thom Popovici** [1,2]
[1] Department of Computing, "Politehnica" University of Timisoara, RO
[2] IeAT, Timisoara, RO
ioana.sora@cs.upt.ro

Motivation

Formal techniques such as model checking and model based testing take as input a model of the system under validation, written in a specific formalism. Usually, such *models are written by hand, based on the system's specifications.*
It would be a big step towards extending the use of formal techniques in practice if such *models could be generated with help of tools directly from the implementations of real systems.*

Proposed approach

We extract behavioral models from the implementation code of real systems, by applying specific **white box techniques** based on the analysis of their **control flow graph**.
The systems to be analyzed are **web applications** and **services**, which can be implemented using many **different technologies**, making code analysis and modeling difficult.
We represent the models as **Extended Finite State Machines**.

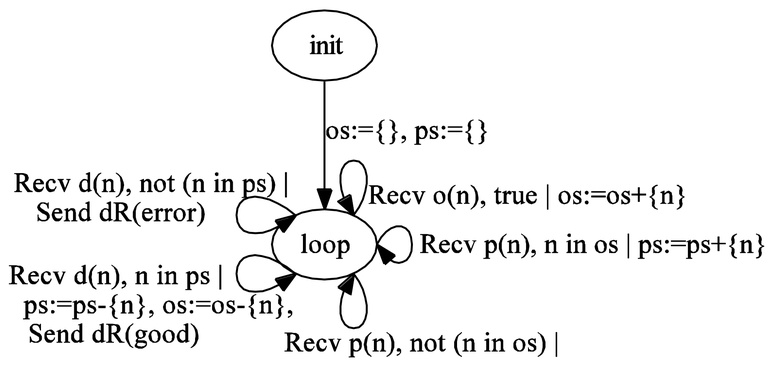
From (abstract) Control Flow Graph to Extended Finite State Machine

Preliminary assumptions:
- The CFG is complete and interprocedural
- There are explicit statements, corresponding to a node in the CFG, for receiving and sending messages of a specified message type and having message parameters.

Transformation principles, in summary:
- Aspects which are *relevant* for the model are those related (data or control) to sent or received messages
- ***"Essential"*** nodes in CFG -> states in EFSM
- a path between CFG nodes which contains at least one "***relevant***" node -> a transition in the corresponding EFSM, with path conditions becoming guards of the transition

Example: Abstract CFG (*here represented as pseudocode* !) of a Shop Server, and its EFSM model

```
1: orders:={}
2: payments:={}
3: while(true)
4:    switch ReceiveMesssage():
5:      case:(orderType, name)
6:          add name to orders
7:      case:(payType, name)
8:          if (name in orders)
9:                add name to payments
10:     case:(deliveryType, name)
11:         if (name in payments)
12                remove name from payments
13:               remove name from orders
14:               SendMessage deliveryResp, goods
15:         else SendMessage deliveryResp, error
16: endwhile
```



Getting the abstract CFG from real service implementations

In practice, web applications and services are developed with the help of special frameworks and APIs. Consequences:
- Instead of explicit statements for sending and receiving messages, frameworks offer complex APIs to describe the interactions of the communicating entities.
- Most often, by analyzing *only* the application code written by the application developer one cannot obtain the whole CFG of the real system (for example server loops are in frameworks)

We implemented **technology specific preprocessing frontends** (until now, for Java RMI, JSP, servlets) which: (1) identify and *abstract the equivalent of send/receive message operations* and (2) *complete the partial CFG extracted from application code to a complete abstract CFG*

Conclusion

In order to cope with the diversity of technologies and APIs which can be used by service implementations, we propose an approach for model extraction in two steps: a technology dependent preprocessing step, followed by a core step that implements a general method of transforming the abstracted control flow graph into an EFSM.
The kind of inferred EFSM is suitable for automatic translation into an entity description in a formal security specification language (such as Aslan++) for distributed systems.

Acknowledgements

This work has been supported by the FP7-ICT-2009-5 project no. 257876 SPaCIoS ("Secure Provision and Consumption in the Internet of Services")