Evaluating the Performance of Software Architectures

D. Petriu (dorin@sce.carleton.ca)

C. M. Woodside (cmw@sce.carleton.ca)

1. Overview

- Introduction
- UCM and LQN
- Correspondences Between UCMs and LQNs
- POTS example
- Conclusion

2. Introduction

- performance analysis should be done early on in the design
- performance analysis is an example of architecture evaluation
- Software Architecture Analysis Method (SAAM) makes use of scenarios as a means of evaluating software architectures
- Use Case Maps (UCMs) are used to illustrate the scenarios and the architecture
- Layered Queueing Networks (LQNs) are used to evaluate the performance

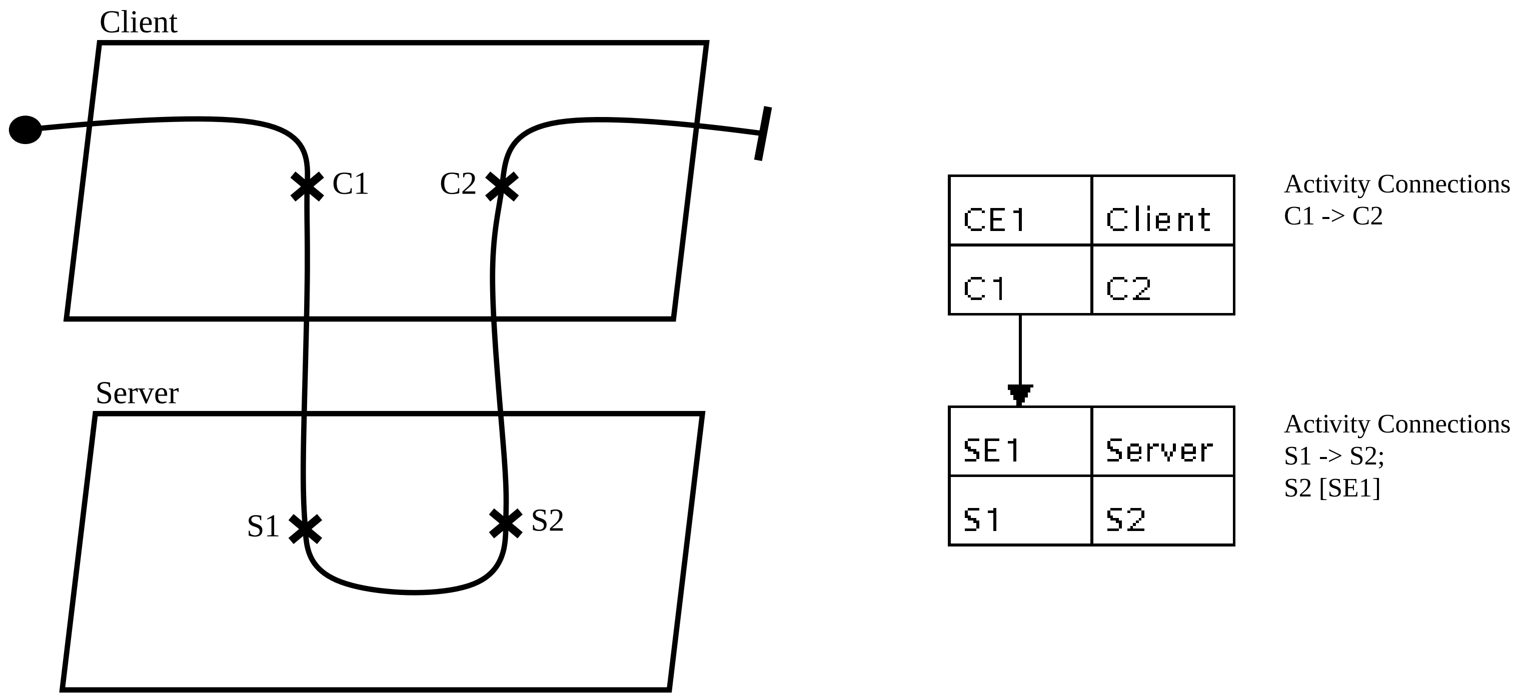
3. UCM and LQN

- UCMs represent scenarios as paths with responsibilities that are executed along the way (may have AND or OR forks and joins)
- UCM paths traverse components that represent system entities
- The architecture of the system is represented by the combination of paths and the way they traverse components
- The UCM Navigator (UCMNav) is a tool used to edit and manipulate UCMs

- LQN models consist of tasks with associated entries and lists of activities
- The tasks are organized in conceptual layers interacting with each other through synchronous calls and returns, or asynchronous calls
- The LQN Solver (LQNS)is a tool that solves LQN models and returns performance parameters for the system
- Jlqndef is a tool that can be used to edit, solve (using LQNS), and display LQNs

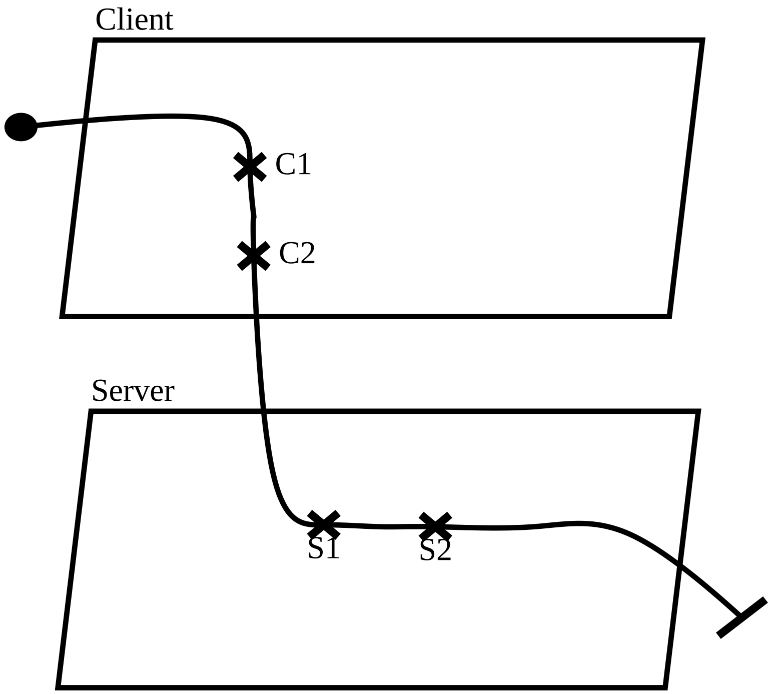
4. Correspondences Between UCMs and LQNs

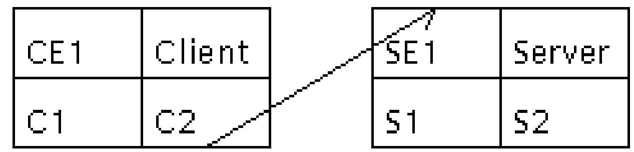
- Synchronous Call and Return



- A synchronous call is made whenever the UCM path crosses from one component to another and returns back to the original component

- Asynchronous Call





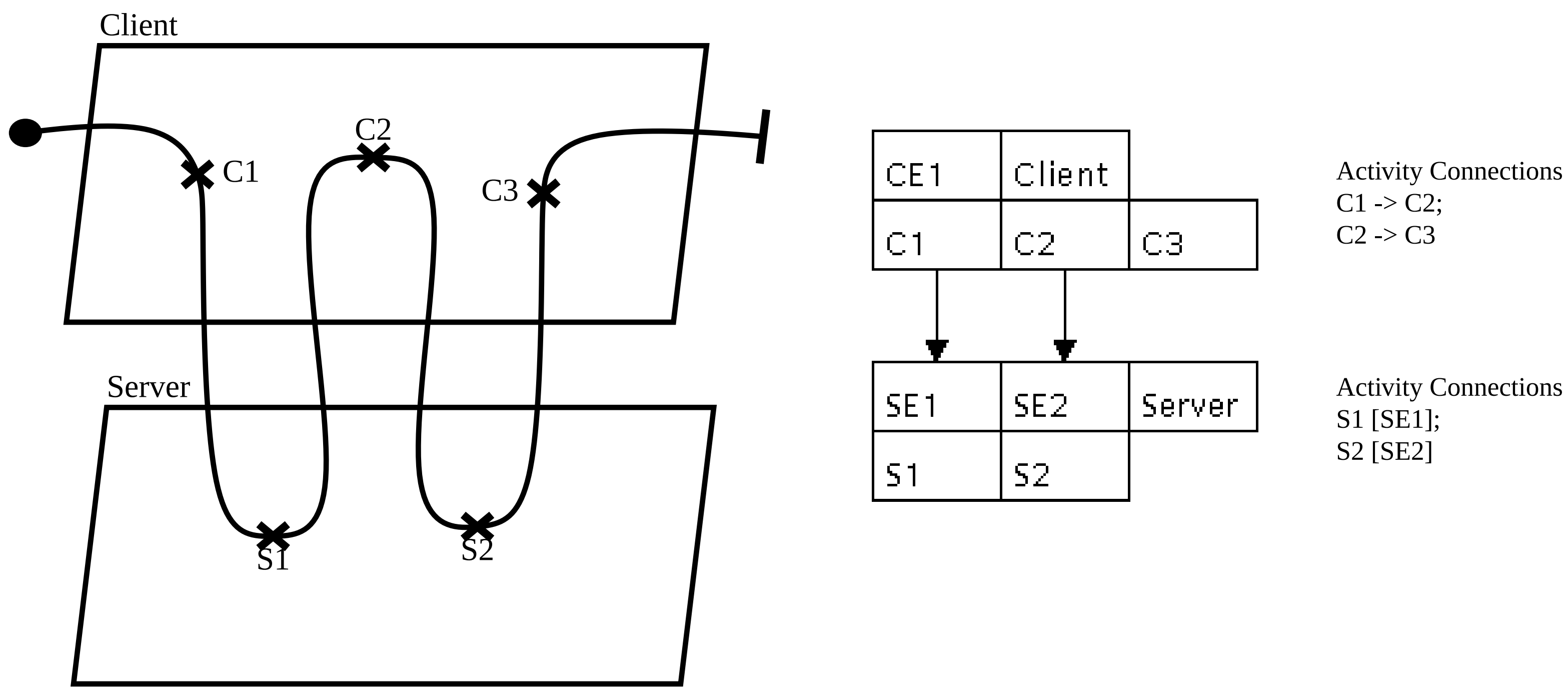
Client Activity Connections
C1 -> C2

Server Activity Connections
S1 -> S2

- An asynchronous call is made whenever the UCM path crosses from one component to another and does not returns back to the original component

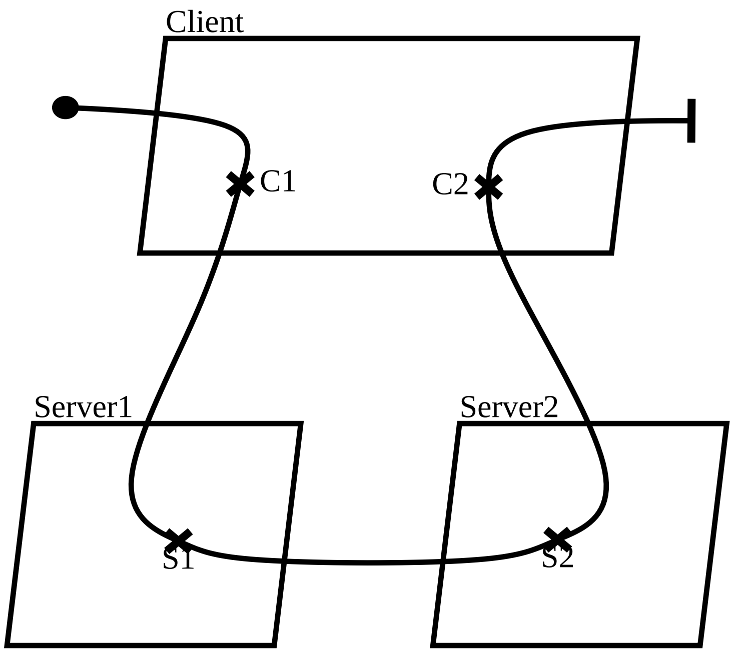
Fig.4: LQN with an asynchronous call.

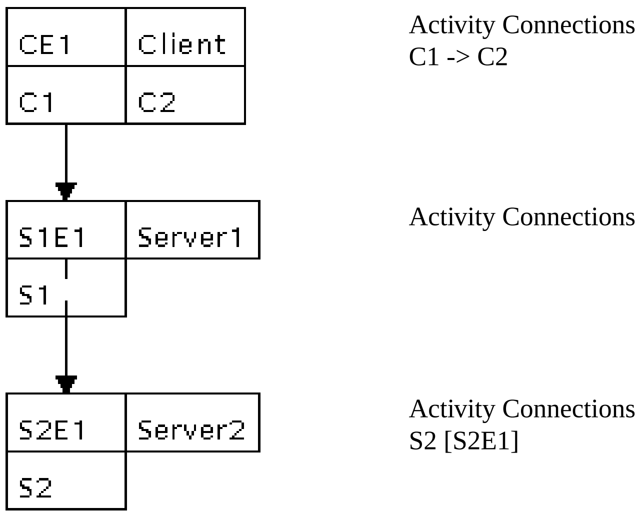
- Multiple Calls



- Multiple synchronous calls are made whenever the UCM path crosses from one component to another, returns back to the original component, and repeats the same pattern

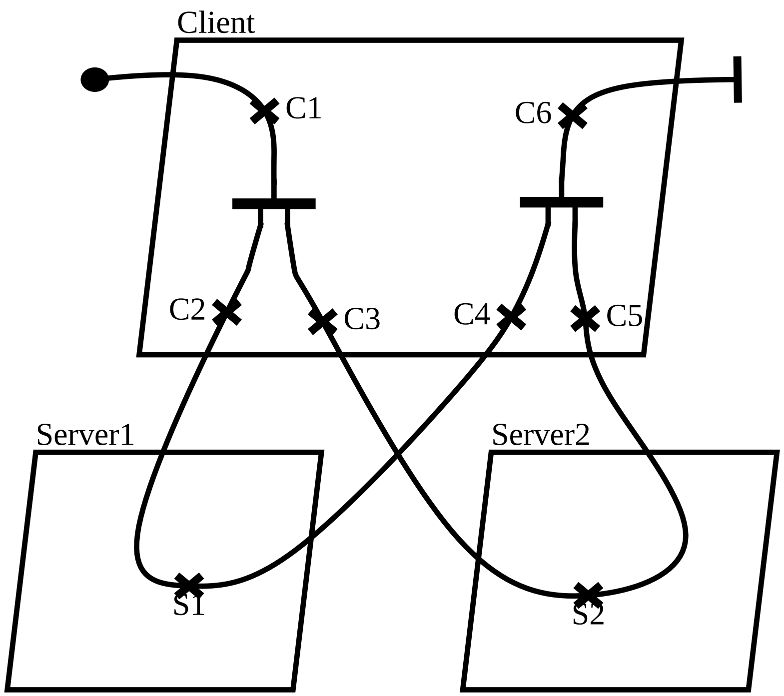
- Forwarding

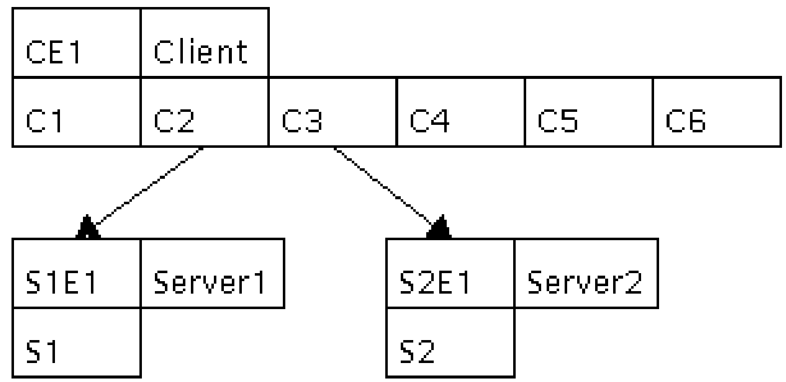




- A call forwarding is made whenever the UCM path crosses from one component to another, and then to several others, before returning back to the original component. The first component makes a synchronous call, but the forwarding is asynchronous for the other components

- AND Fork and Join





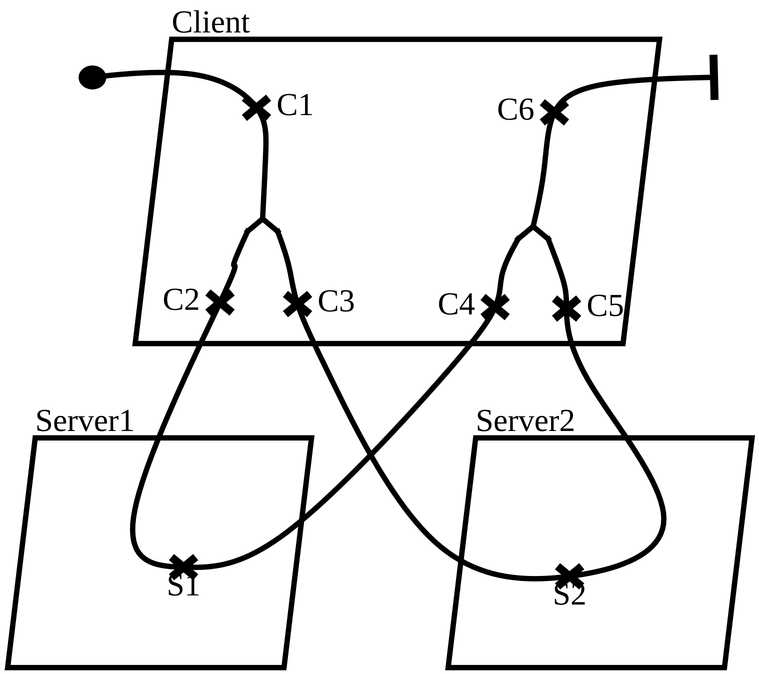
Client Activity Connections
C1 -> C2 & C3;
C2 -> C4;
C3 -> C5;
C4 & C5 -> C6

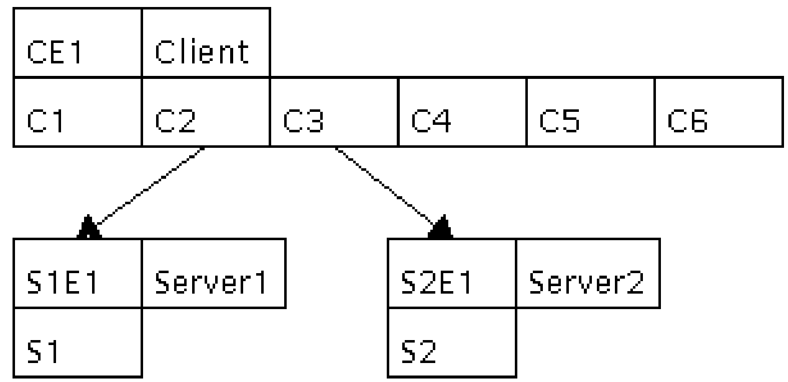
Server1 Activity Connections
S1 [S1E1]

Server2 Activity Connections
S2 [S2E1]

- An AND fork and join are put in the calling component. By making two synchronous calls after the AND fork, parallel services are triggered in the other components.

- OR Fork and Join





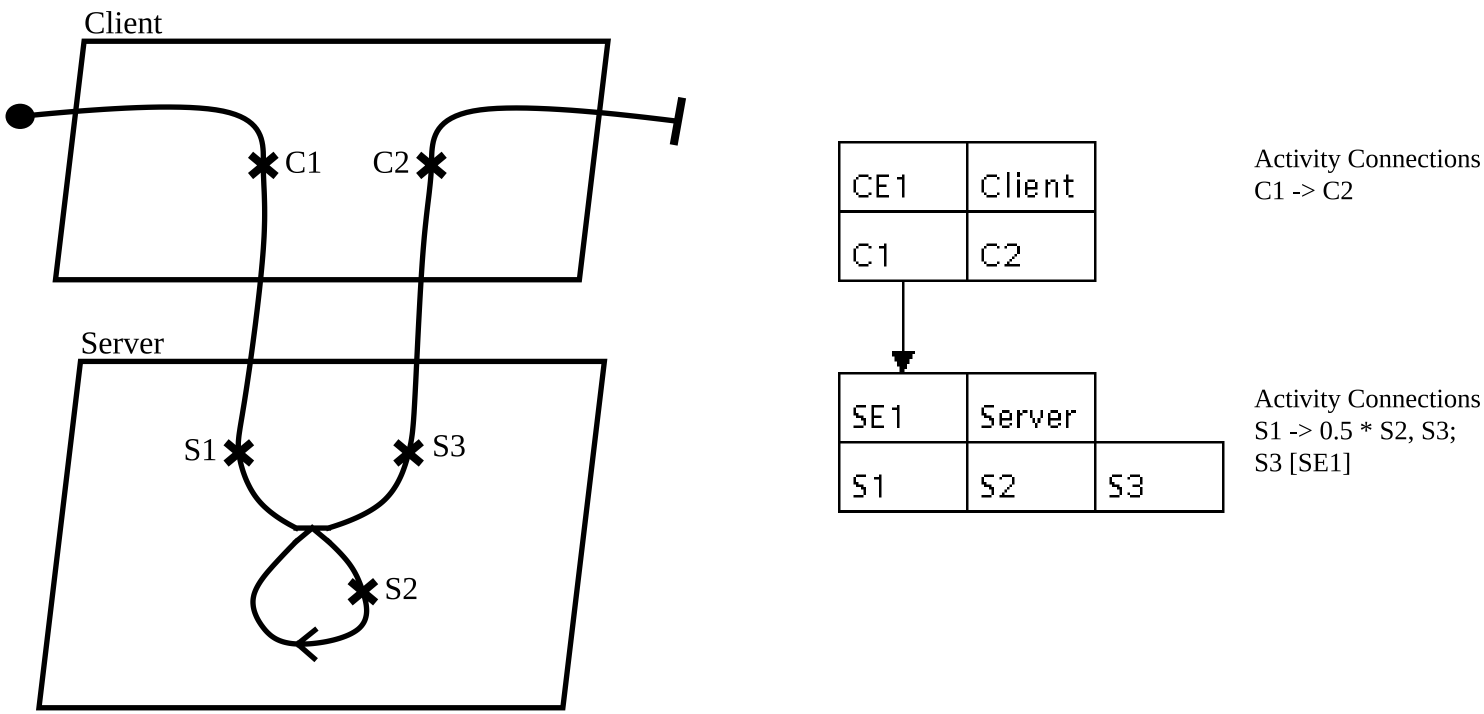
Client Activity Connections
C1 -> (0.5) C2 + (0.5) C3;
C2 -> C4;
C3 -> C5;
C4 + C5 -> C6

Server1 Activity Connections
S1 [S1E1]

Server2 Activity Connections
S2 [S2E1]

- An OR fork and join are put in the calling component. By making two synchronous calls after the fork, competing alternate services are triggered in the other components.

- Loop

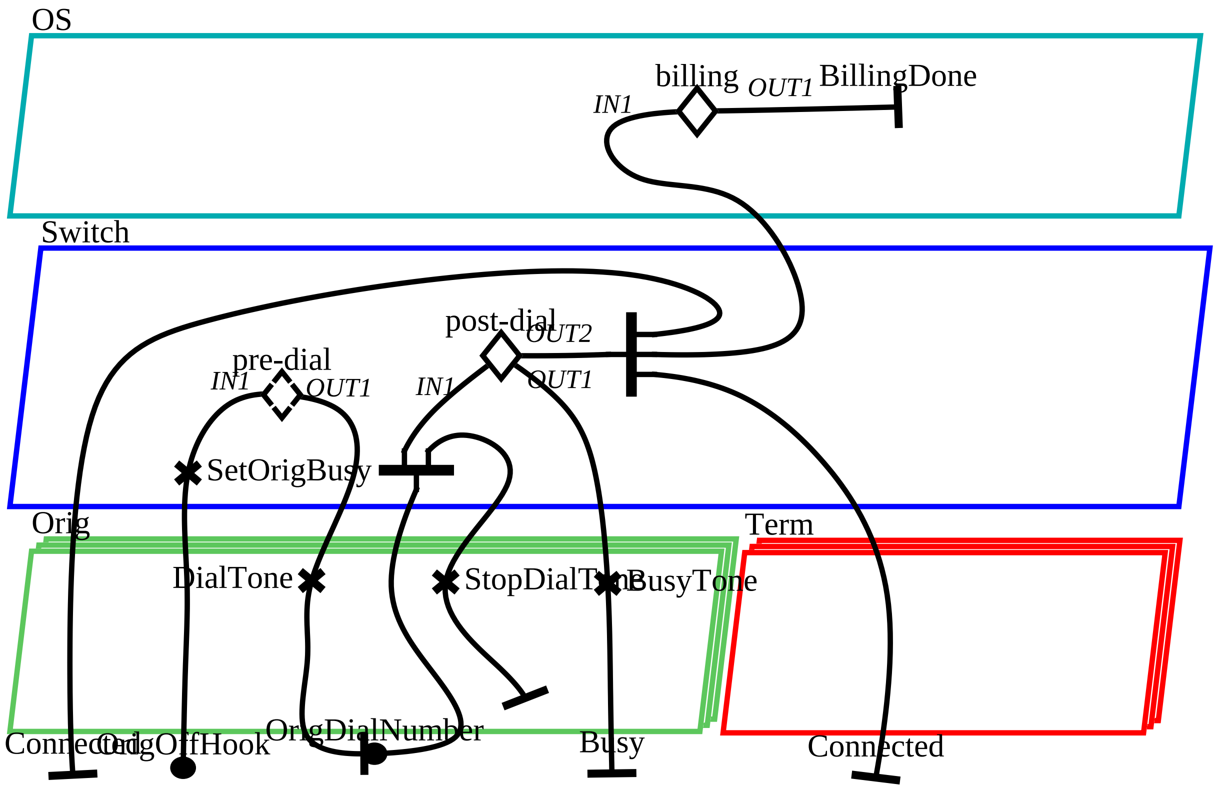


- A loop is indicated by a special UCM loop construct that appears the same as an OR join followed immediately by an OR fork.

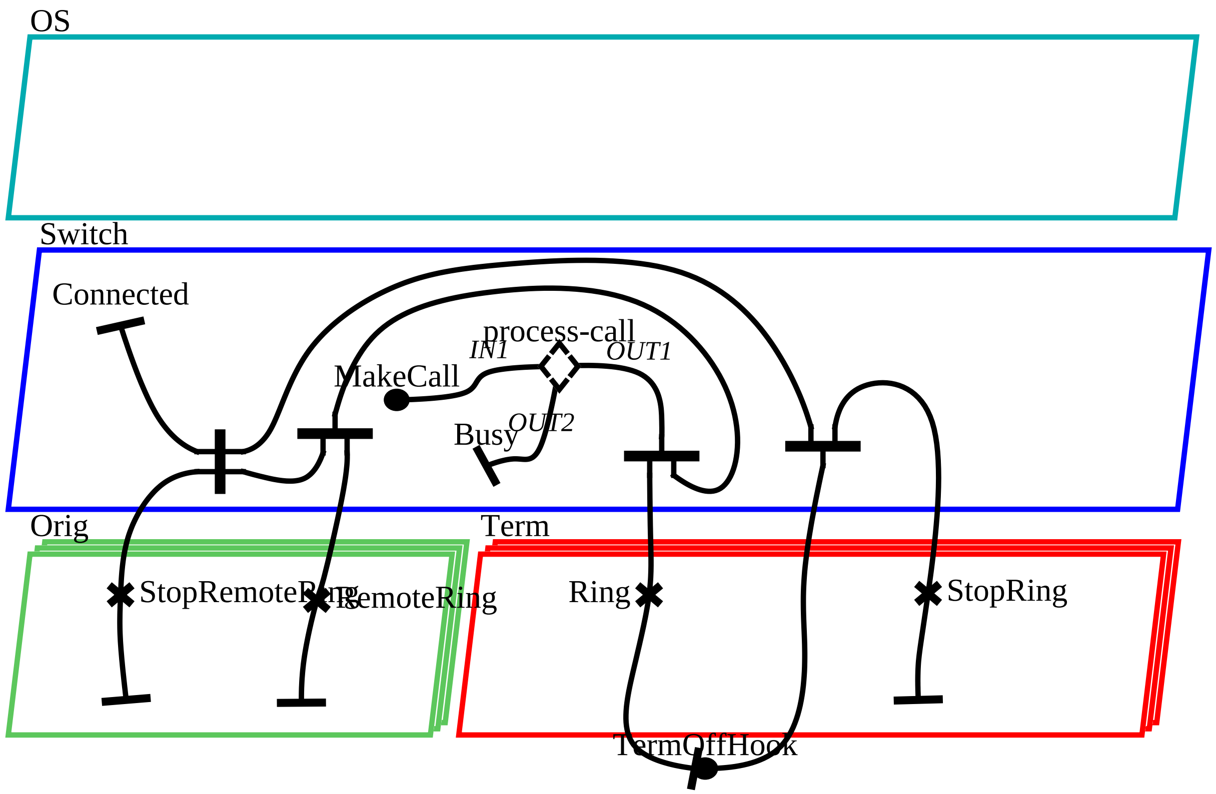
5. POTS Example

- based upon the POTS functionality described in the Feature Interaction (FI) Detection Contest as part of the 5th International Workshop on Feature Interactions.
- The components on the map are as follows:
  - **Orig** - process corresponding to the call originator’s (caller) telephone or telephone device
  - **Term** - process corresponding to the call terminator’s (callee) telephone or telephone device
  - **Switch** - process corresponding to the service provider’s telecommunications switch
  - **OS** - process corresponding to the service provider’s operations system server

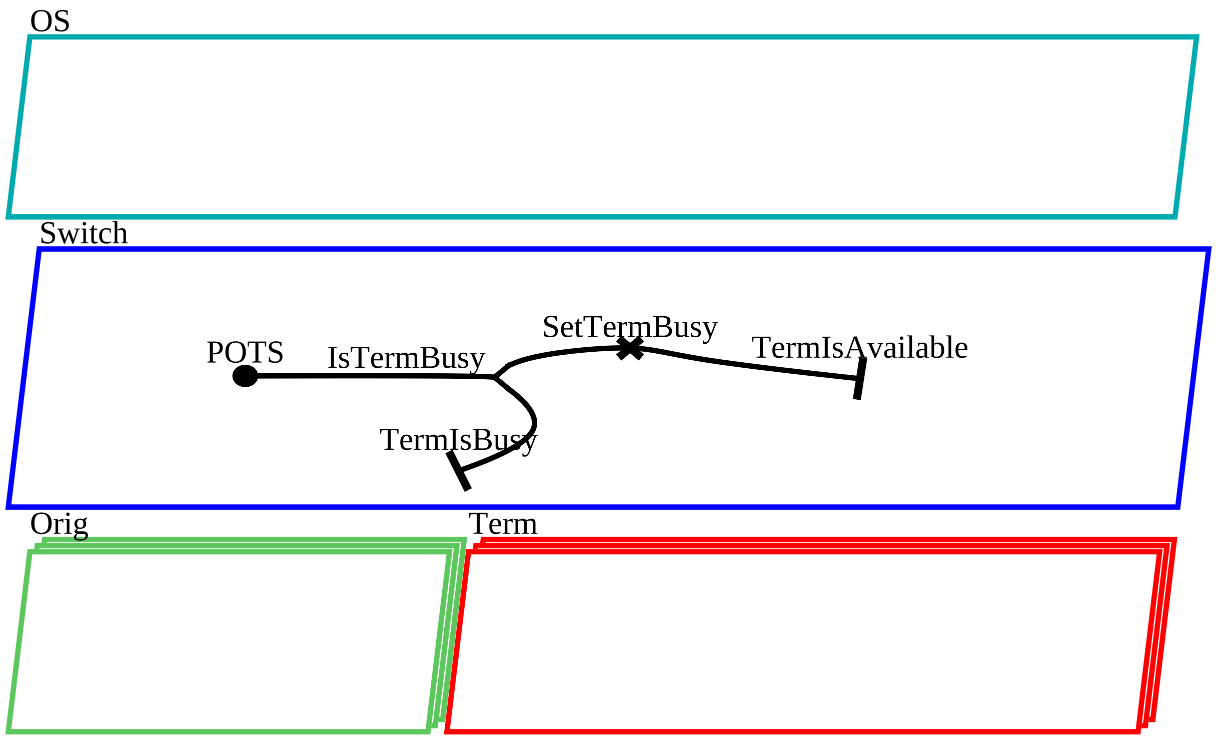
- POTS Root Map



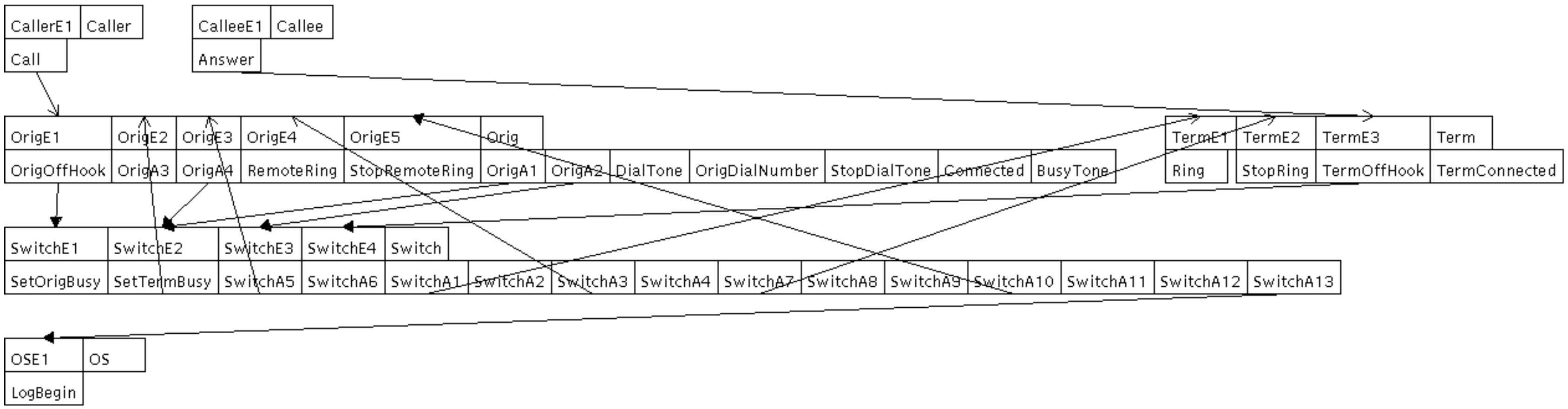
- POTS Post-Dial Plug-in shows how POTS works after the number has been dialed and until the call has been established.



- POTS Process-Call Plug-in encompasses the essential call processing logic of the telephone system.



- POTS LQN model



6. Conclusions

- We have demonstrated an effective way to bring performance analysis to the early software development stages.
- Our framework for transforming UCM designs into LQN performance models can be applied across a wide range projects.
- The next step in our project is to finish implementing a UCM2LQN tool that will automatically convert UCMs from the UCMNav into LQNs