ECLIPSE Feature Logic Analysis

presenter: Greg Bond (AT&T Labs-Research)

joint work with: Franjo Ivancic (Univ. of Pennsylvania),
Nils Klarlund, and Richard Trefler (AT&T Labs-Research)

Summary

- a tool for checking behavioral properties of features (services) developed for the ECLIPSE service architecture e.g. call waiting
- ECLIPSE is an architecture for managing feature interaction in IP telecom, based on the DFC architecture

Overview

- Motivation
- Strategies
- DFC
- Inter-Port Messaging and Protocols
- Feature Logic Analysis
- Model generation
- Model checking
- Conclusions

Motivation

- To support feature interaction management
  - Features in ECLIPSE are modular
  - Features are inter-connected using the pipes and filters architectural pattern
  - Features communicate with one another using pre-defined protocols

Motivation (cont'd)

- Because features are arranged in a pipeline, the integrity of a call is dependent upon integrity of the individual features
- Experience programming ECLIPSE features has revealed that it is easy to violate protocols

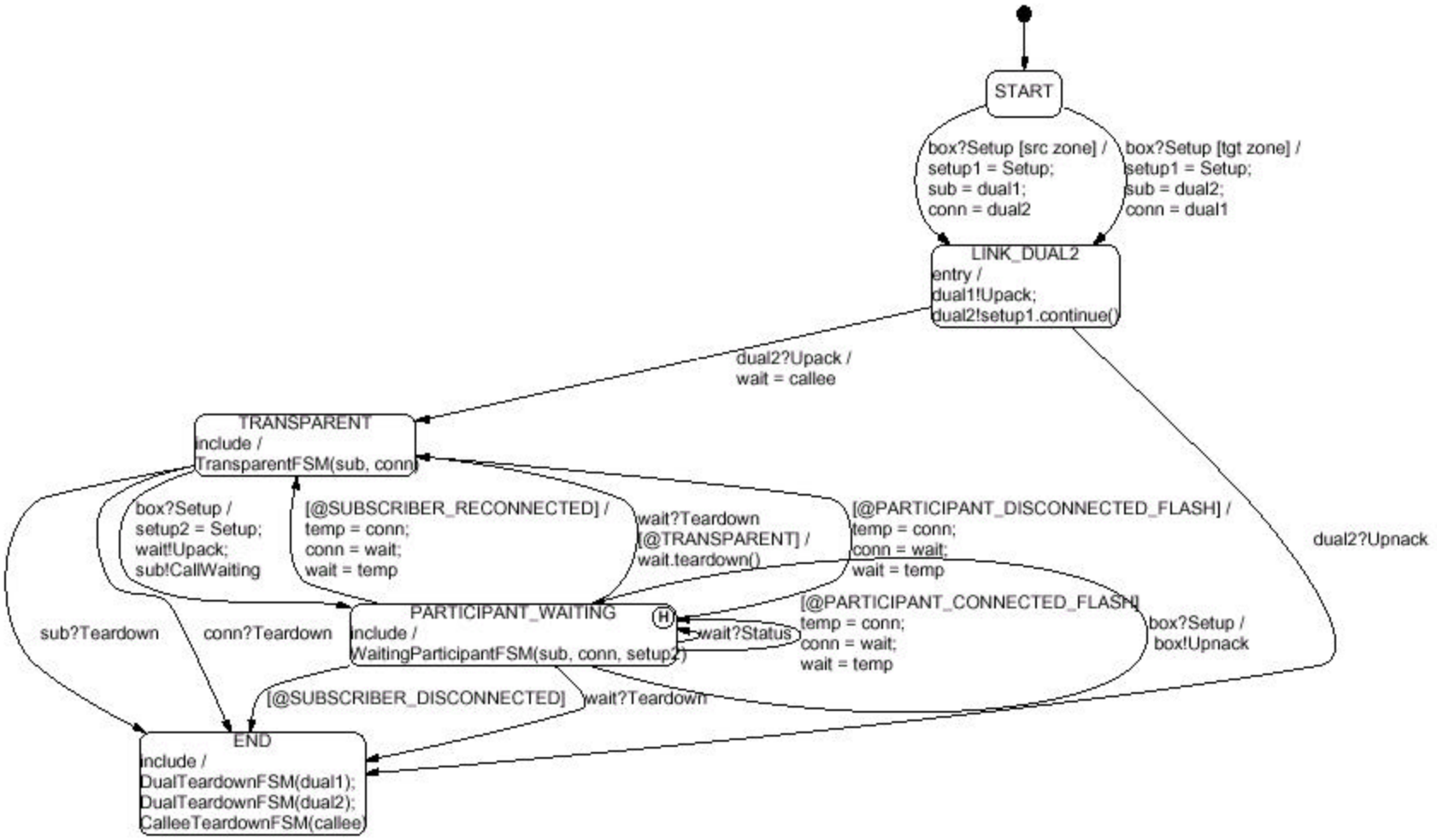
Strategies

- Run-time monitoring to ensure that protocols are satisfied
- Disadvantages
  - Run-time overhead
  - Protocol violations only detected after feature deployment

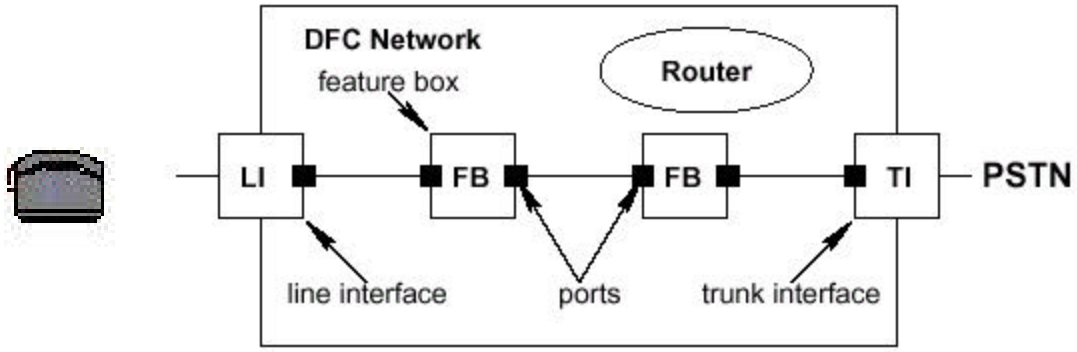
Strategies (cont'd)

- Domain-specific language, ECLIPSE Statecharts, that extends UML Statecharts for ECLIPSE feature design and implementation
- Advantage is clear mapping from design to implementation which helps expose logic errors early in development
- Also amenable to formal automated analysis using model checking
- ECLIPSE Statecharts currently translated to Java

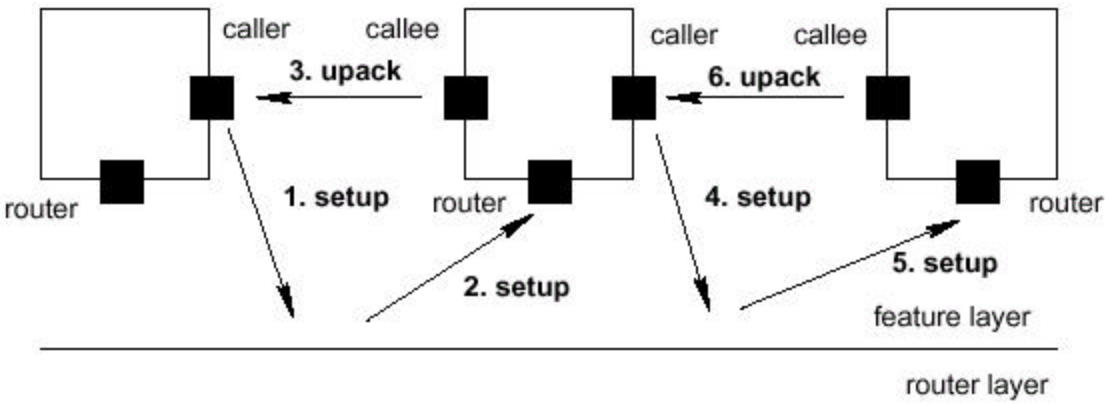
Call Waiting Feature Logic



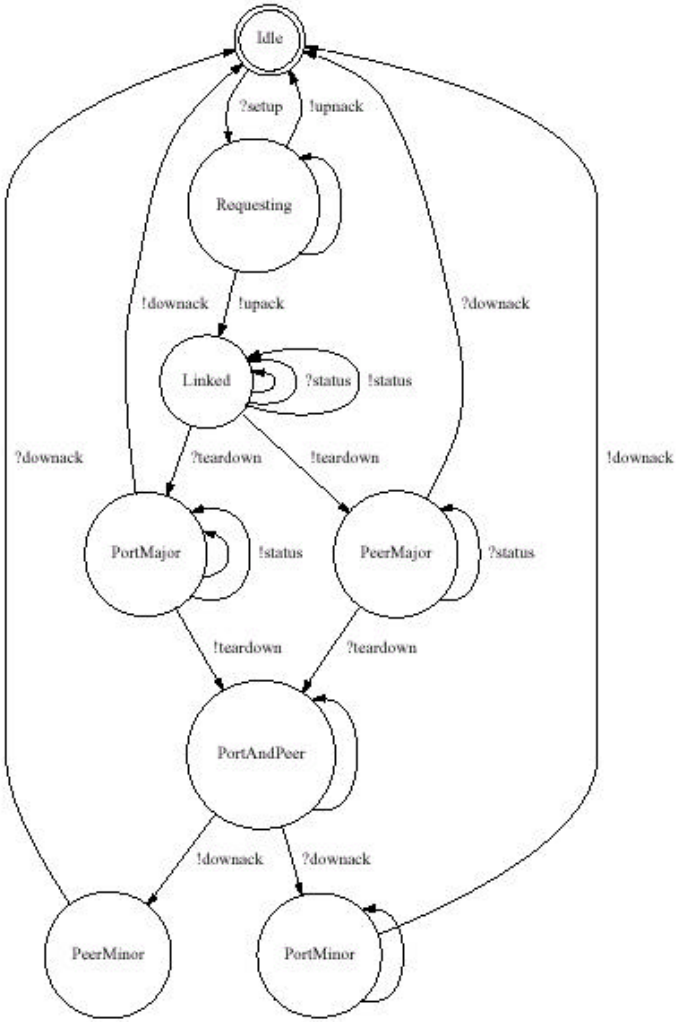
DFC


DFC Network
feature box
Router
LI
FB
FB
TI
PSTN
line interface
ports
trunk interface

Inter-Port Messaging and Protocols



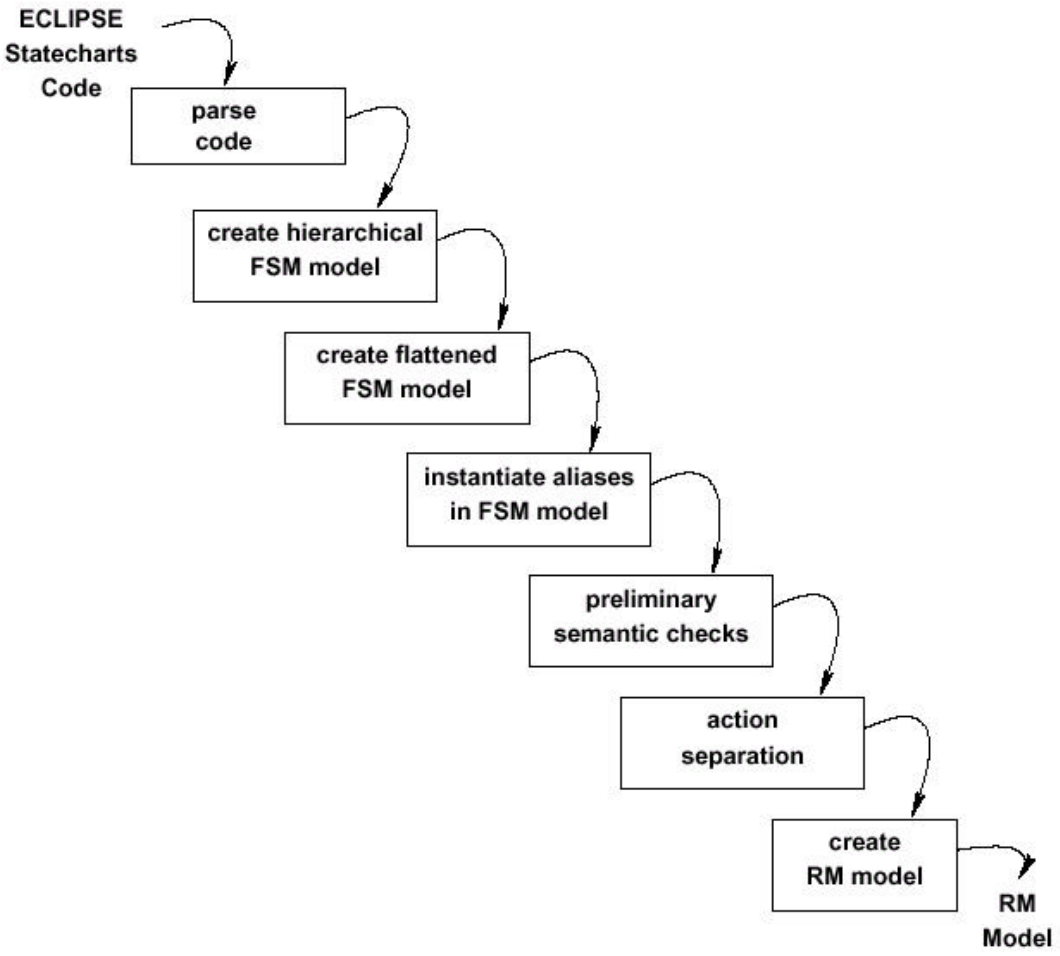
Caller Port Peer Protocol



Feature Logic Analysis

- Feature logic analysis addresses the following two questions:
  - Did the programmer of the feature box consider all possible input messages that the environment—the peers associated with the feature box—can send to the feature box?
  - Does the feature box output only those messages to its environmental peers that are expected by those peers?

Model Generation



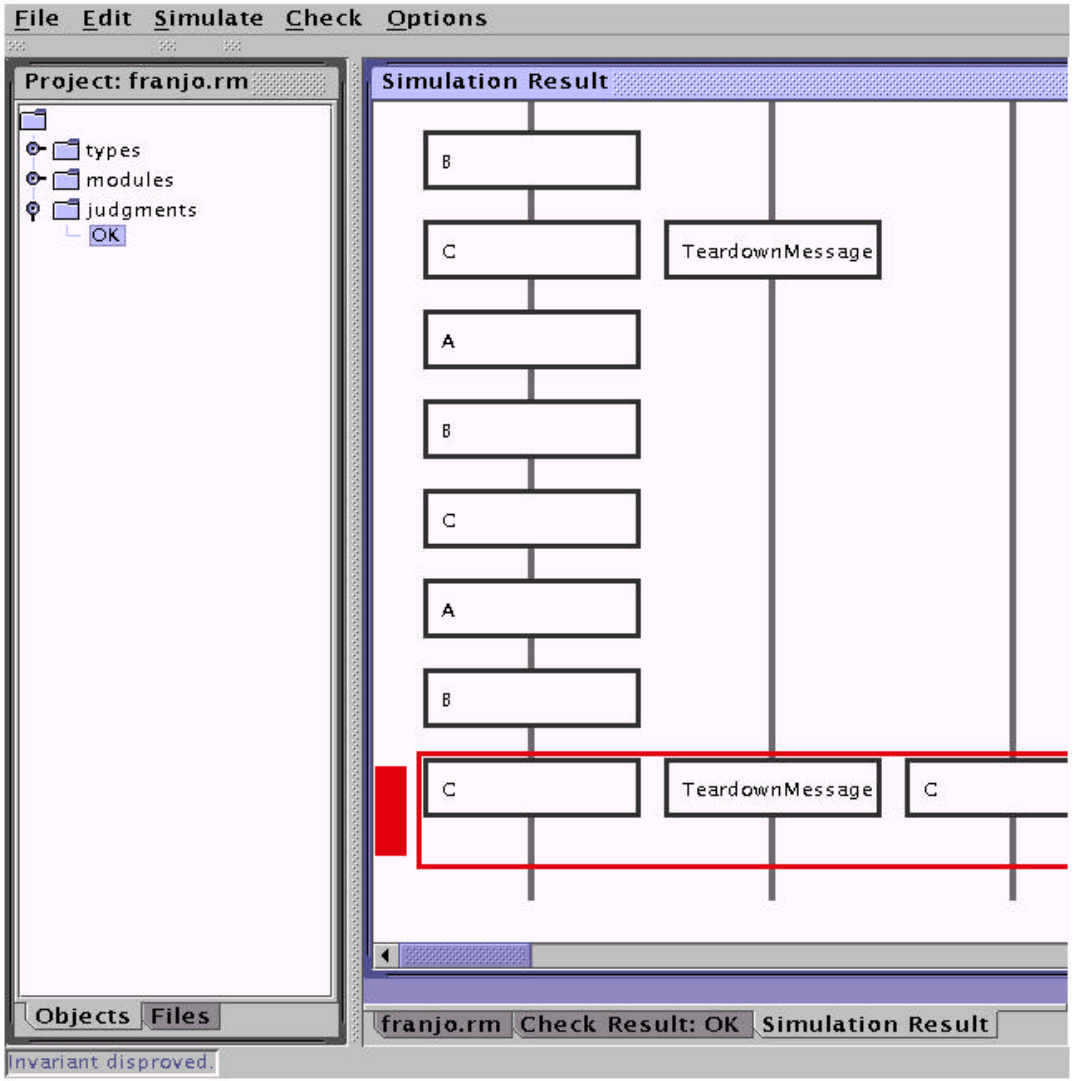
Reactive Modules Model

- Reactive Modules (RM) language is a modular temporal logic (alternating temporal logic) for specifying reactive concurrent systems
- Mocha tool (Univ. of Pennsylvania) is able to efficiently verify invariants for state space defined by an RM specification

Model Checking

- Translated flattened FSM captures operational semantics
- Default transitions to a “bad state” added for messages received
  - by box program that box program not programmed to accept
  - by peer protocol model that peer protocol not specified to accept
- Model + peer protocol models submitted to Mocha model checker
- Mocha checks that “bad state” not reachable

Bad State Analysis



Analytical Correctness

- No formal proofs of completeness or soundness
- Both proofs require formal description of
  - ECLIPSE Statecharts semantics
  - semantics of translation
  - RM semantics

Related Work

- A number of results exist relating to model checking various subsets of Statechart dialects
- Distinguishing characteristic of our approach is its applicability to unbounded queues between ports and their peers
- Our approach exploits characteristics of DFC peer port protocols and is not applicable for any peer port protocol

Future Work

- Determining general properties of peer protocols that permit abstracting away queues
- Extending approach to support latest multimedia protocol extensions to ECLIPSE
- Characterizing semantics of ECLIPSE Statecharts

More Information

- Pamela Zave's IPtel presentation tomorrow afternoon
- http://www.research.att.com/projects/eclipse