Engineering & Computer Science

Concordia UNIVERSITY

A Requirement Level Modification Analysis Support Framework

Maryam Shiri

Concordia University
Montreal, Canada

Motivation

- Support management and decision makers during the early modification analysis.
  - Determine impact of a modification request.
  - Estimate the test cases that have to be retested.
  - Identify Use case scenarios, which contain feature interactions.

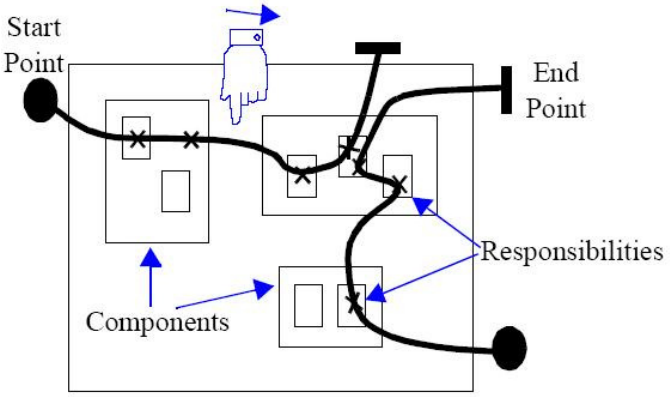
Approach

- Based on Use Case Maps (UCM)
- Utilizing Formal Concept Analysis (FCA)

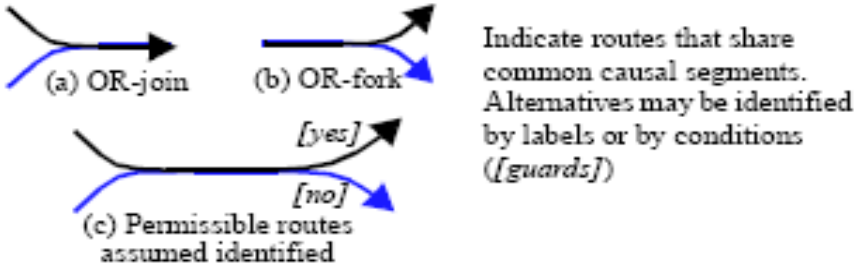
Use Case Maps (UCM)

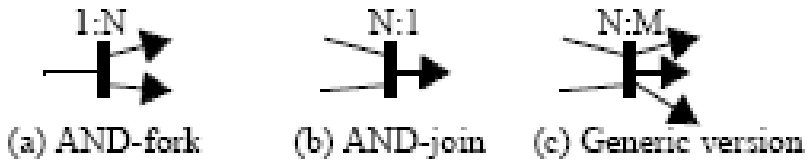
- A graphical **scenario** notation (map-like diagram) developed [R.J. Buhr 1995]
- A notation, for expressing scenarios above the level of messages exchanged .
- UCM allows developers to model dynamic systems where scenarios and structures may change at run-time.
- Facilitate moving towards design
- UCM part of URN (User Requirement Notation, Being standardized by ITU-T in Z.15x)

UCM Basic Notation



- Alternative/Concurrent behaviour

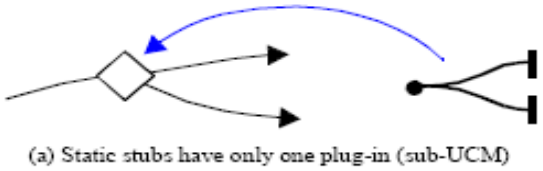


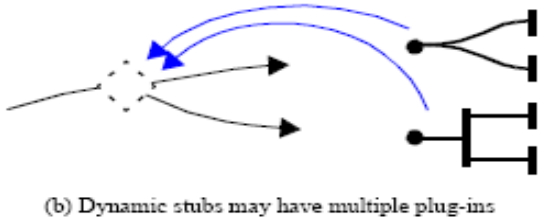


Shared routes and OR-Fork /Joins

Concurrent routes with AND-Fork/Joins

- Model dynamic (run-time) refinement for variations of behaviour and structure





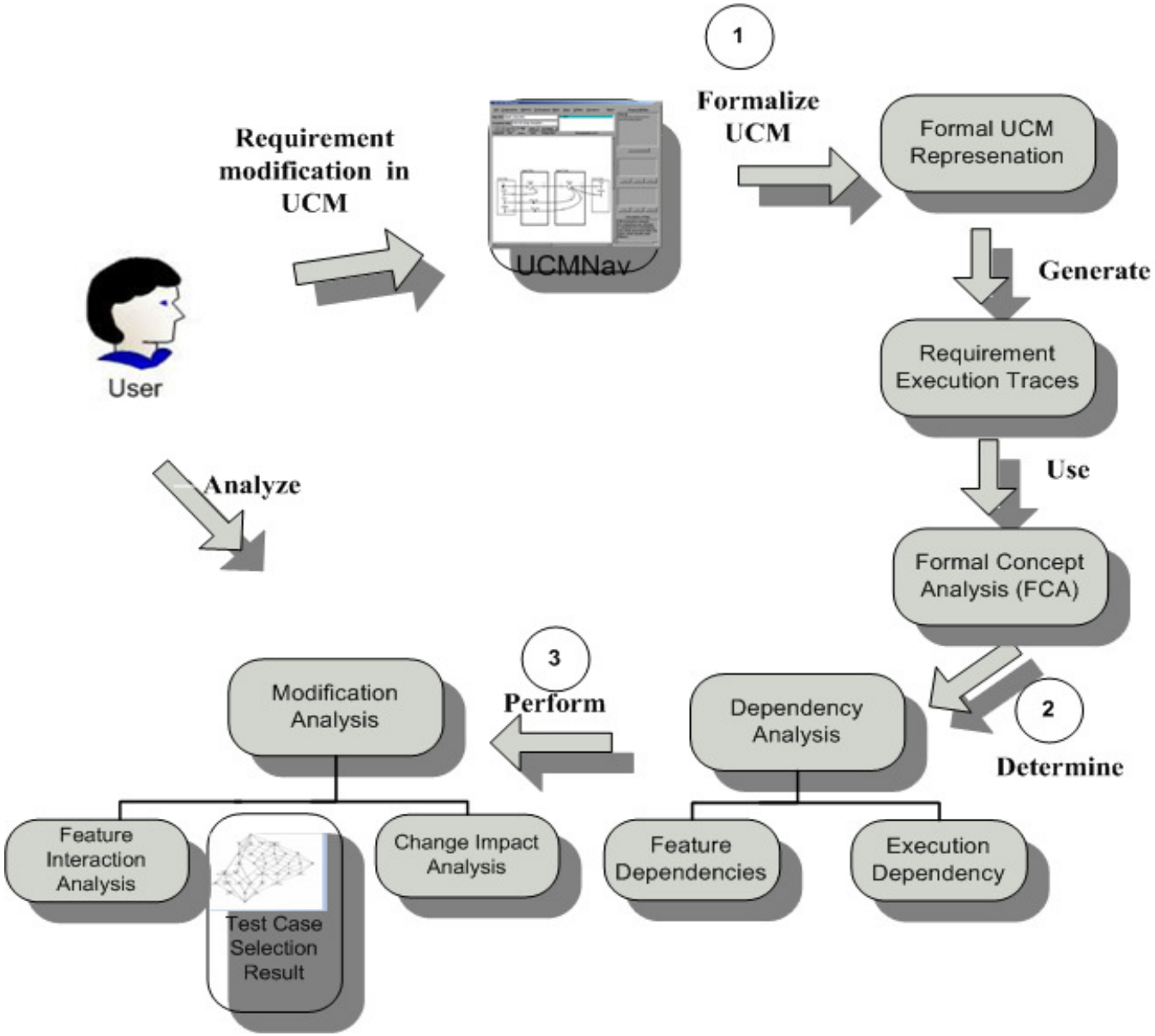
Static stubs have only one plug-in

Dynamic stubs may have multiple plug-ins

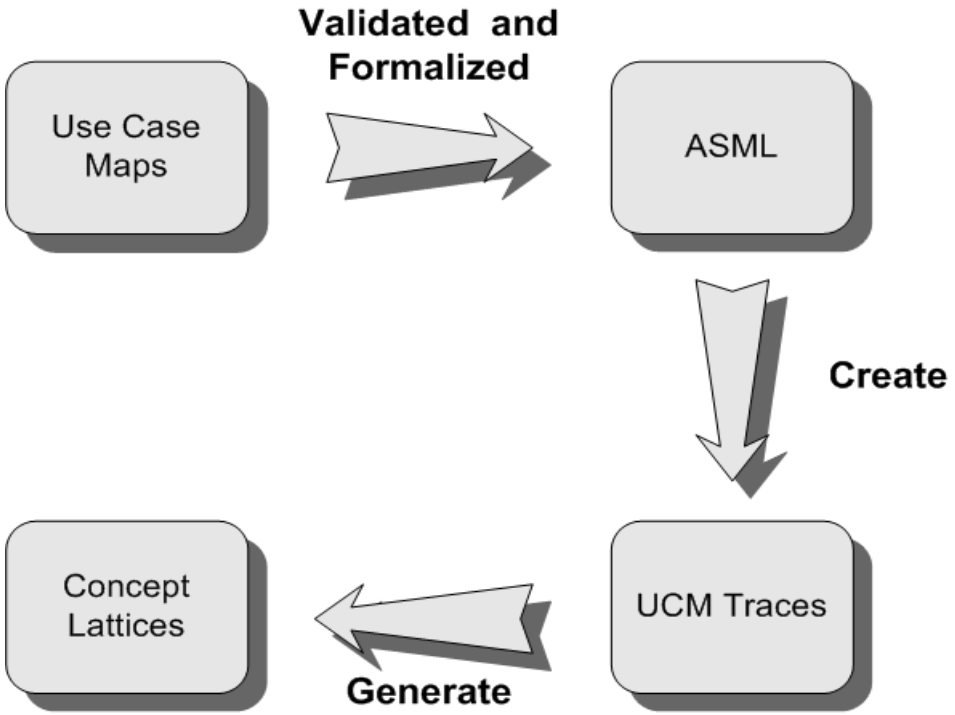
Formal Concept Analysis

- Formal Concept Analysis (FCA) - Ganter & Rudolf Wille [1982] - is a data analysis technique based on lattice theory – Birkhoff [1967].
- FCA can perform a **sensible data grouping**
- FCA is gaining acceptance and has been applied to various application domains
  - Software Engineering: Module Restructuring, Feature-driven program understanding, etc.

Requirement Modification Analysis



UCM => FCA

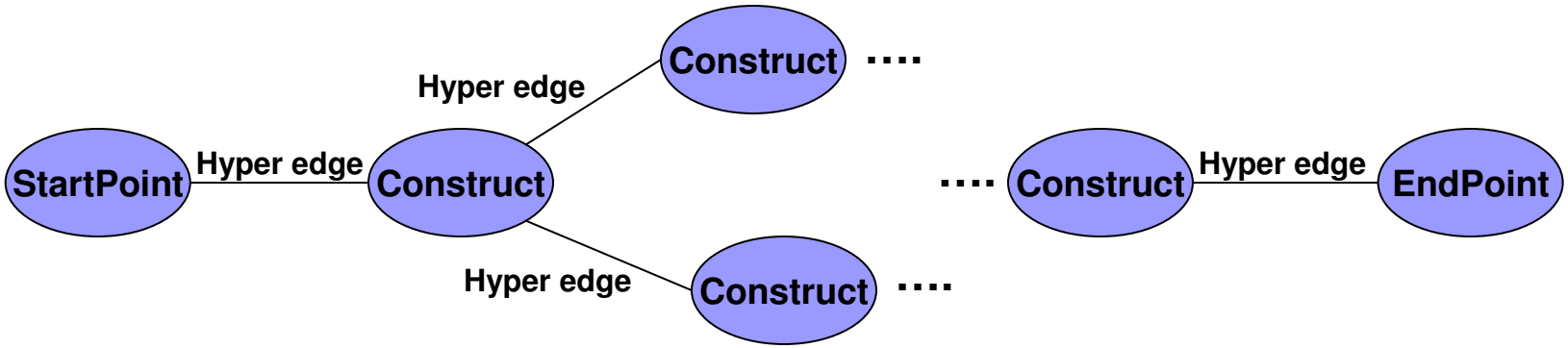


⇒ *Formal Semantics for UCM*

⇒ *Generate traces*

⇒ *Dependency analysis on UCM traces (functional, execution dependencies) through FCA*

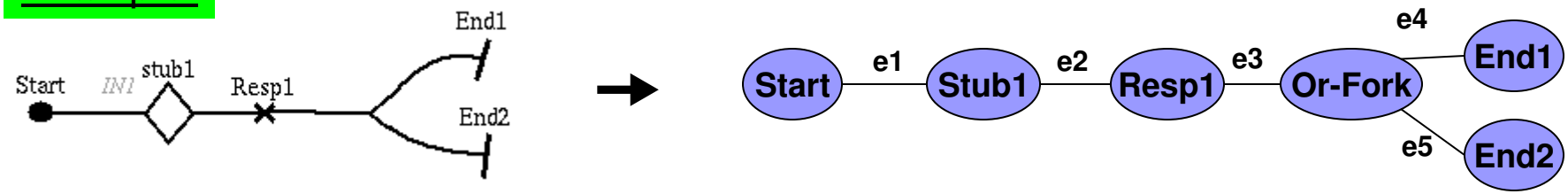
Formalizing UCM - UCM Hyper Graph



: Hyper edge

: UCM construct (Start Point, End Point, Or-Fork, And-Join, ….etc.)

Example:



UCM= {(Start,e1,Stub1), (Stub1,e2,Resp1), (Resp,e3,Or-Fork), (Or-Fork,e4,End1), (Or-Fork,e4,End2)}

UCM Trace Example

```
Start Executing: MainReq
MainReq.active:in1

MainReq:Rule of Start Point:Req
MainReq.active:e1

MainReq:Rule of Start Point:Start
MainReq.active:O1
Responsibility: deny in component: Agent_Term
MainReq.active:OCS5

MainReq:Rule of End point:fail
MainReq:UpStub executed
MainReq.active:e4
```

Dependency Analysis on UCM Traces

- **Functional Dependency**
  - Two or more scenarios contain the same sub-scenarios, (share sub-scenarios with the same start and end points) or
  - Two Scenario Sc1 and *Sc2* are functional dependent if they share the same plug-in (feature) *Pa*, where

$$P = \{Pa, Pb \ldots Pz\}$$

Execution dependencies

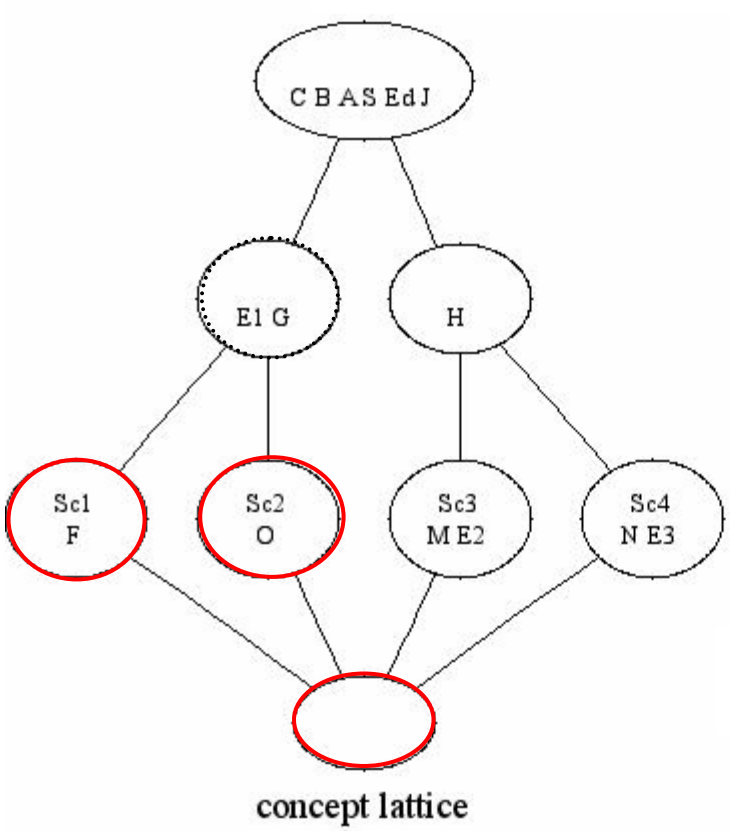
- **Component execution dependency**
  - From a UCM perspective a component dependency exist, when two or more scenarios share the same component C',

    where $C' \subset$ {set of UCM components}.

- **Domain element execution dependency**
  - We can state that E is a set of UCM domain elements
  - Domain elements are execution dependent if their scenarios share common Use Case Map domain elements.

Change Impact Analysis

- Predicting Direct Impacts



concept lattice

Sc1: A S B C Ed J F G E1

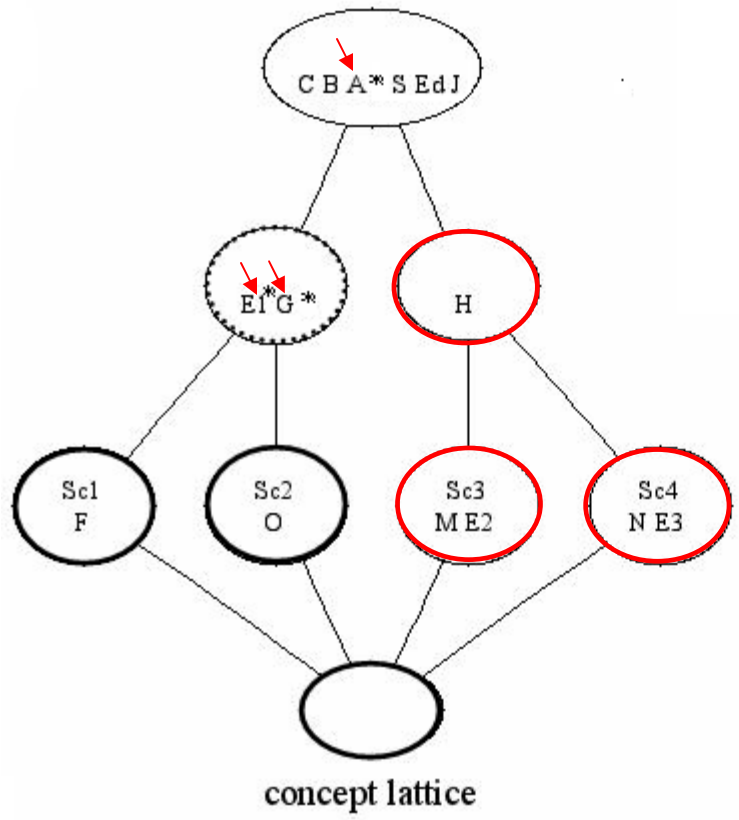
Sc2: S B C Ed J O G A E1

Sc3: A S B C Ed J H M E2

Sc4: A S B C Ed J H N E3

Change Impact Analysis Cont'd

- Predicting Indirect Impacts
  - Enrich concept lattice with notation of sequence



Sc1: A S B C Ed J F **G E1**

Sc2: S B C Ed J O **G A E1**

Sc3: A S B C Ed J H M E2

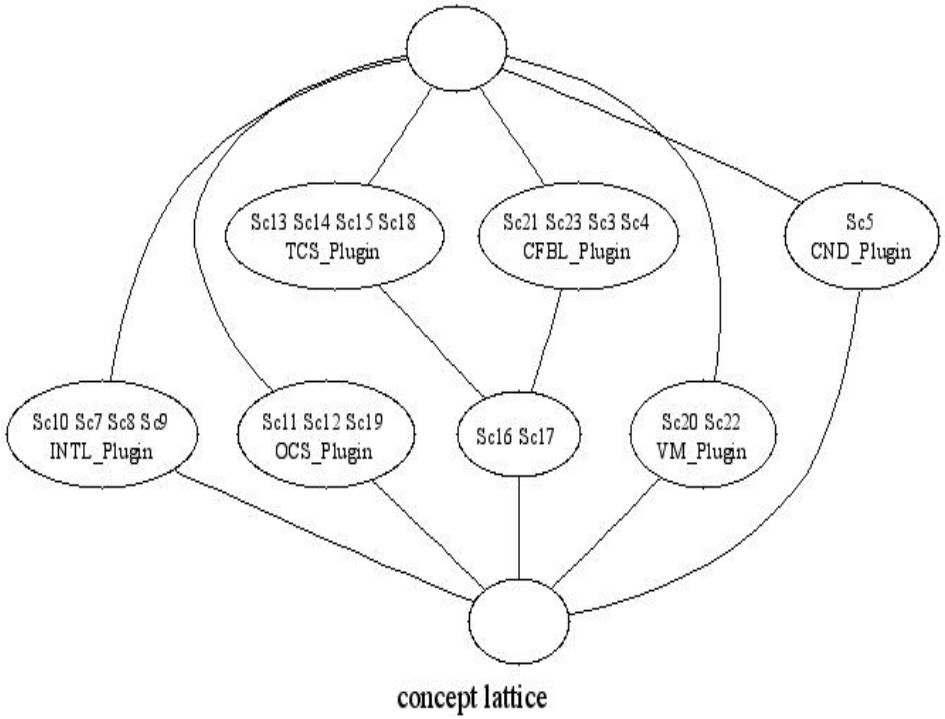
Sc4: A S B C Ed J H N E3

Predictive Regression Test Selection

- Starting from the node to be modified, possible test cases to be retested are identified by traversing the execution dependency lattice downwards to identify all the reachable leaf node (scenarios).

Feature Interaction Analysis

- Step1 - Execution traces are generated and collected for all scenarios defined in the UCM
- Step 2 - FCA is created and those scenarios with no or only one feature can be eliminated (*FI never occurs*)
- Step 3 - All attributes from the upper levels in the concept lattice that are associated with a particular scenario are passed down, remaining scenarios are classified (*FI occur* or *FI prone)*

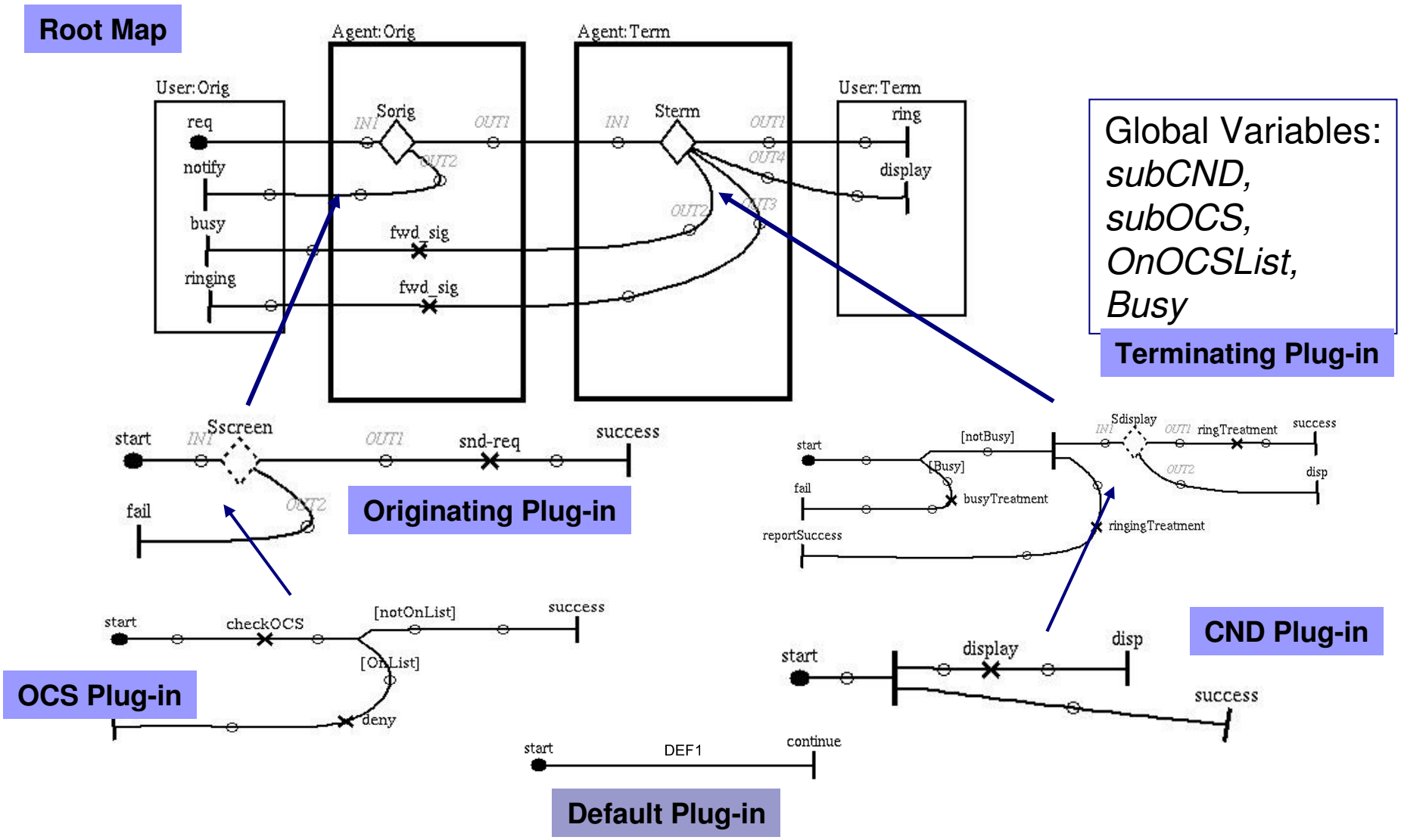


concept lattice

UCM with FCA

- FCA depends on the quality and coverage of traces.
- We assume UCM scenario coverage that is, every scenario at the UCM level was executed at least once.
- Depending on the dependency an appropriate object, attribute pair can be selected.
- The resulting FCA can then be visualized as a context lattice through visualization tools.

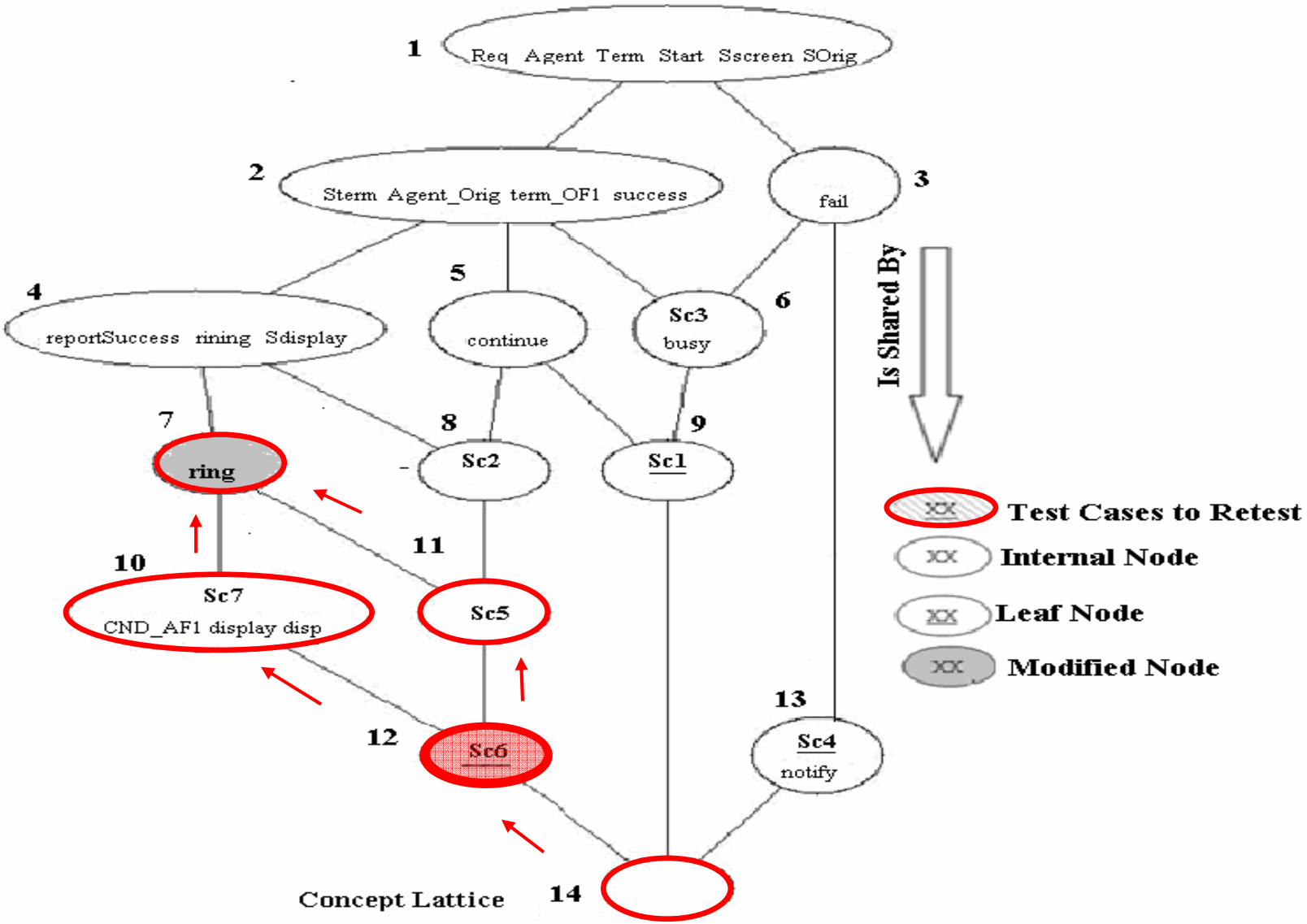
Case Study: A Simple Telephony System



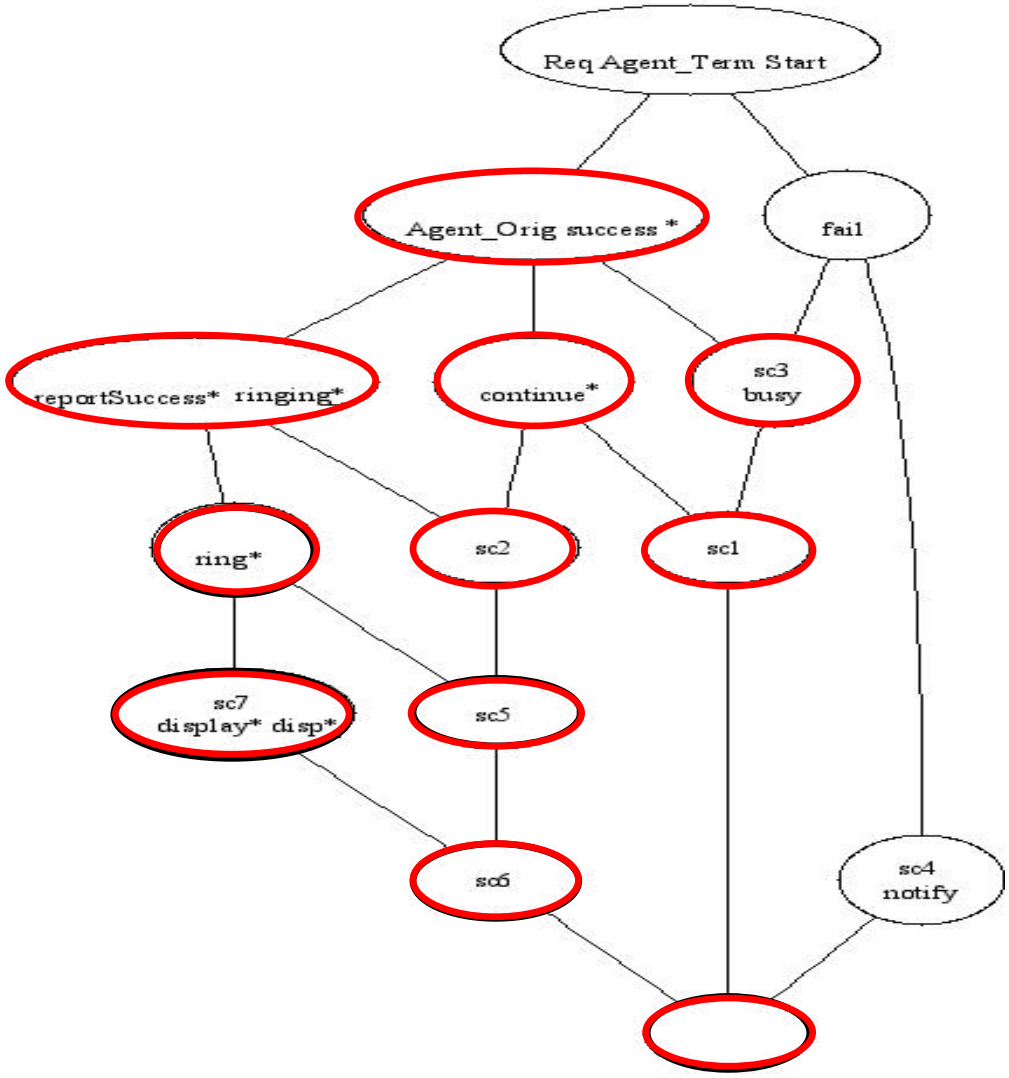
System Scenario Definitions

Scenario Group	Number	Scenario Name	Variables			
			Busy	OnOC-SList	subCND	subOCS
Basic Call	1	BCbusy	T	-	F	F
	2	BCsuccess	F	-	F	F
OCS Feature	3	OCSbusy	T	F	F	T
	4	OCSdenied	F	T	F	T
	5	OCSsuccess	F	F	F	T
CND Feature	6	CNDdisplay	F	-	T	F
OCS_CND	7	OCS_CNDdisplay	F	F	T	T

Change Impact Analysis and Selective Regression Testing



Indirect Impact



concept lattice

Conclusions and Future Work

- Combined UCM + FCA and applied them for modification analysis at the requirements level.
- Tool support for the analysis
- As part of our ongoing work,
  - larger case studies have to be performed to further validate our approach .
  - Consider notion of time in UCMs (timed UCMs).
  - Consider a threshold technique to filter some of the indirect impacts
- Enhance analysis by addressing some of FCA's limitations
  - Lack of sequence
  - Lack of data and control dependencies