A Domain Specific Language for Contextual Design

HCSE 2010, Reykjavik, Iceland

Balbir Barn and Tony Clark

Middlesex University

b.barn@acm.org

Middlesex University

Structure of the talk

- Introduction
- User centred design and Model driven development
- Motivation:
  - Experiences of UCD – Case study outcomes
  - The case for modelling in UCD
- The Central question
- Contextual Design
- A model driven language engineering approach
- A DSL for Contextual Design
- Concluding remarks

In a nutshell

- UCD processes and artifacts are ambiguous and lack precision. Even the the more “model” based methods do not have sufficient semantics.
- A model driven approach to language design is proposed and Contextual design models such as “Cultural Models” are given a language treatment to support the development of bespoke tools.

User centred design and Model driven development

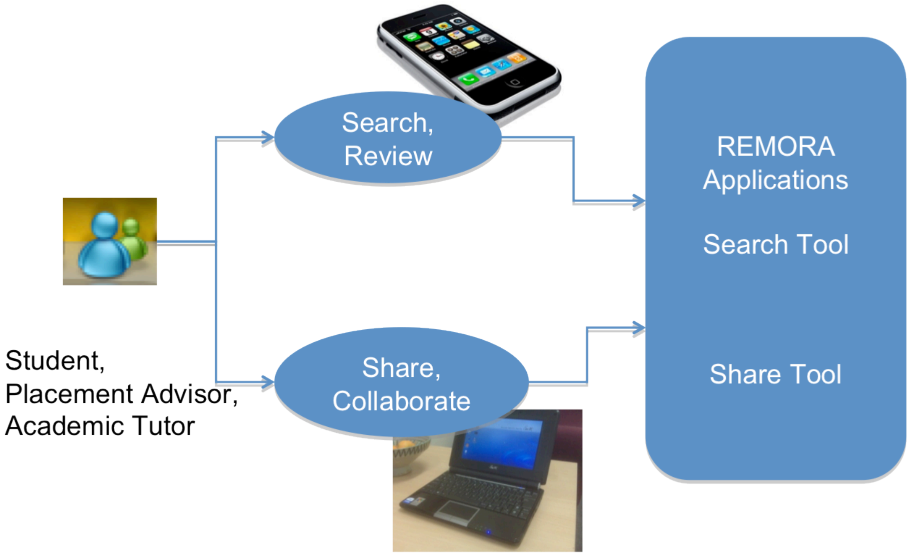
- UCD
  - Users as equal partners in the design process - but involving users can present problems
  - HCI and SE: A cultural gap
- SE practice evolving towards model driven development (MDD)
  - MDD – offers greater affordance to address representation gap between understanding and implementation
  - a greater focus on precision
  - Support for multiple viewpoints and transformations between viewpoints
- Recognition of tension between:
  - Lack of precision of UCD one side
  - and alienation of users in MDD approaches

Motivation

- A recent experience with UCD: The Remora project
- Key problems arising from UCD
- Could model driven approaches help?

Motivating case study: Remora

- Aims
  - to provide mobile software applications to support work-based learning and assessment for social workers "in the wild"
- Objectives
  - Build software tools that students and social workers want and need - using a user-centred approach to elicit requirements
  - Evaluate tools and their usage to provide key knowledge to inform JISC E-Learning Strategy



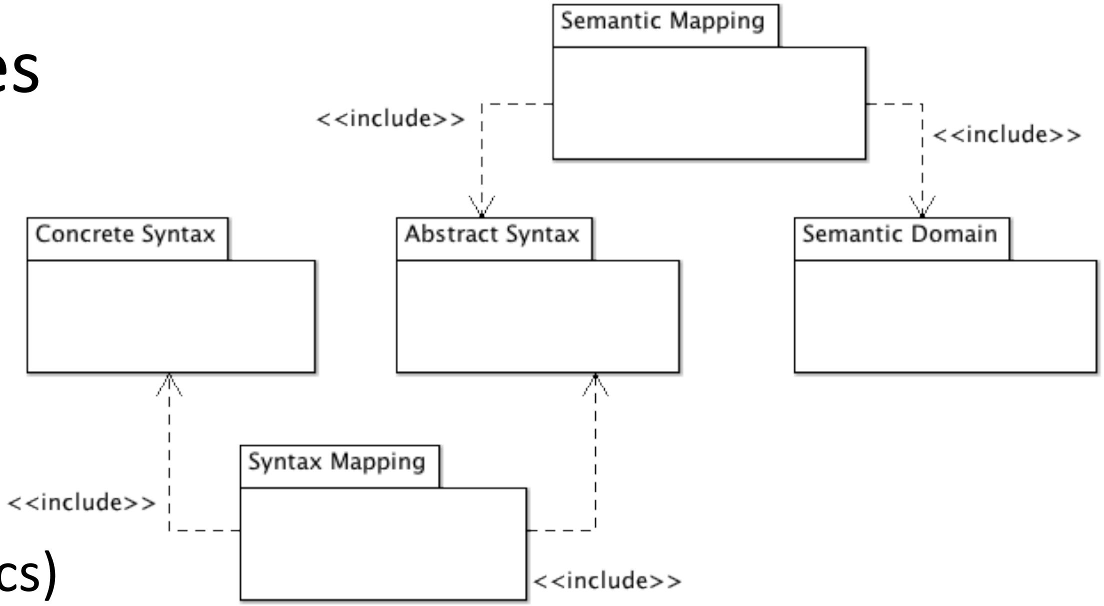
- Experiences with UCD
- Multi-disciplinary team, Multiple development locations
- Multiple approaches to development
- Move towards a co-design methodology

Key problems

- User types
  - An application that goes across multiple device types and has different user types of differing experience
- Users as designers
  - Users can have important and relevant ideas but they are not designers
- New technologies
  - Many new emerging technologies – users do not have knowledge to understand the entire ecology of technology
- Work environments
  - Limited knowledge at management level
  - Work pressures
- Deployment risk
  - Fear of coping with technology
  - The profession of Social Work is high risk
  - Precautionary risk – data security.
- User confusion of what they want and what they need

A model driven language engineering approach

- Model driven principles
- Language definition
  - Concrete syntax
  - Abstract syntax
  - Semantic domain
  - Mappings (syntax, semantics)



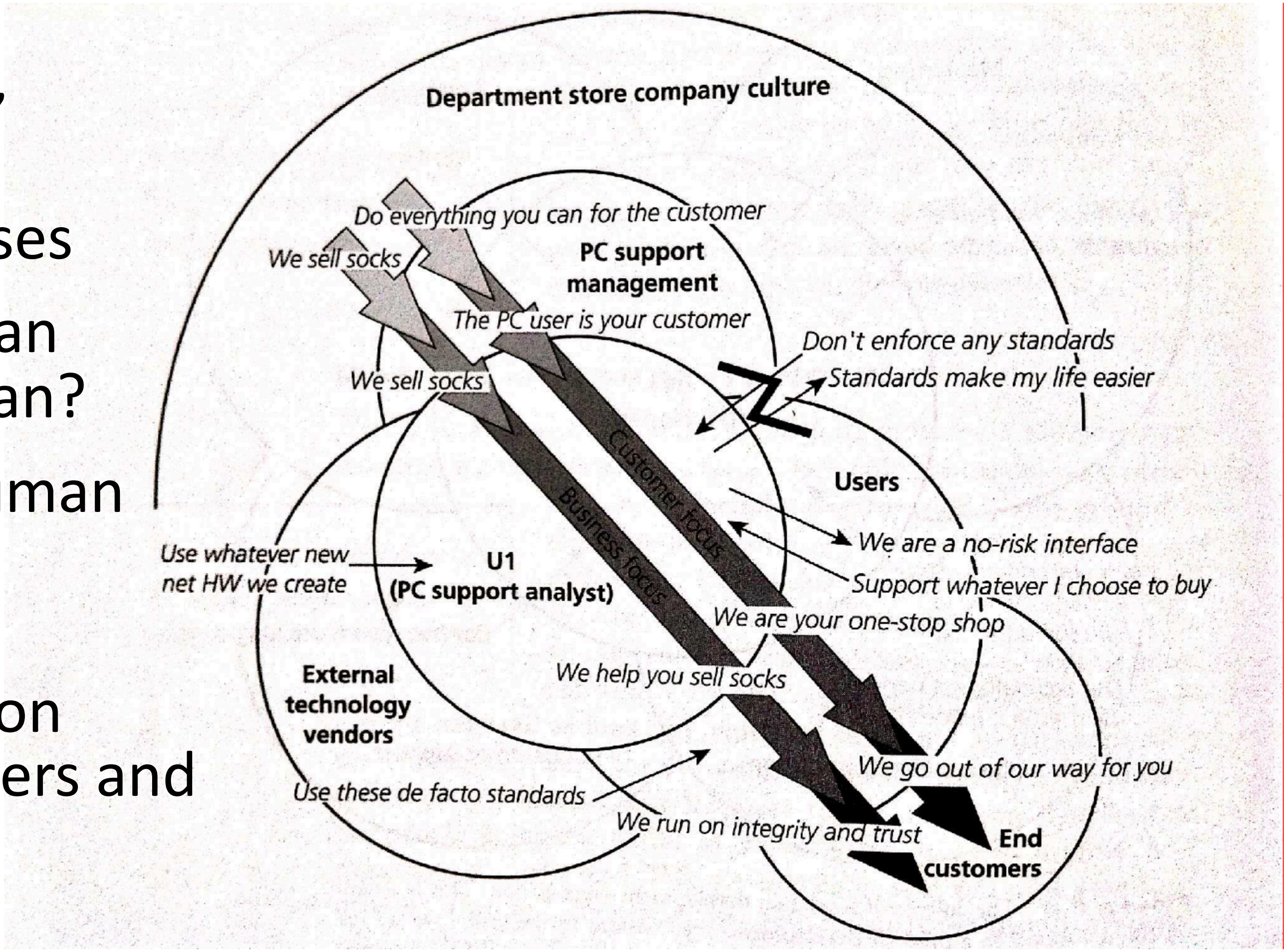
- Meta modelling language
  - For the abstract language – it can be UML.
- Tooling

Contextual Design

- Contextual Design (Beyer and Holzblatt 2001)
  - Rich in UCD and has affinity with SE approaches
- Focus on artifacts, where and how work is done; intuitive elements of the environment.
- Subset of key models include:
  - Artifact model, Flow Model, Sequence Model and Cultural Model

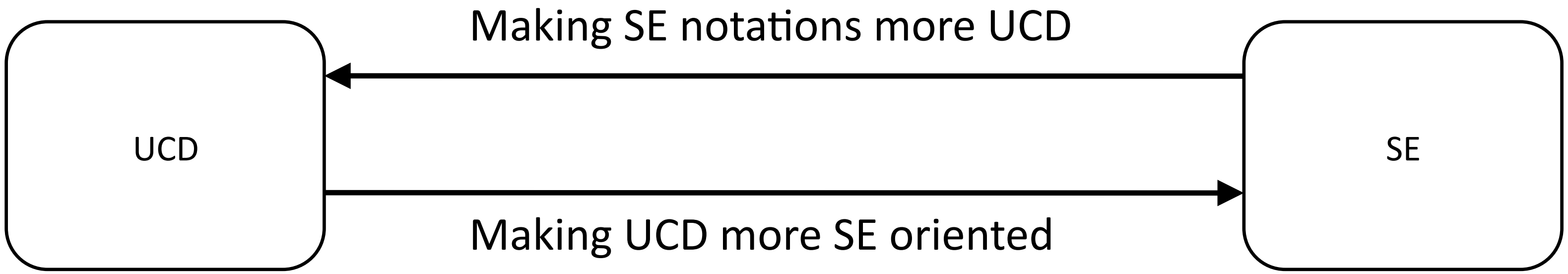
Semantics of Cultural Models

- Arrow sizes, directions
- Size of ellipses
- What does an overlap mean?
- Requires human analysis
- Issues of interpretation between users and designers


Department store company culture
Do everything you can for the customer
We sell socks
PC support management
The PC user is your customer
We sell socks
Don't enforce any standards
Standards make my life easier
Customer focus
Business focus
Users
We are a no-risk interface
Support whatever I choose to buy
Use whatever new net HW we create
U1 (PC support analyst)
We are your one-stop shop
We help you sell socks
External technology vendors
We go out of our way for you
Use these de facto standards
We run on integrity and trust
End customers

A case for modelling UCD

- UCD is strong on user engagement but the artifacts cannot be easily transformed to support multiple viewpoints
  - Design slicing
- Model based Artifacts make transformations between viewpoints possible
  - In design and design-implementation



Making SE notations more UCD

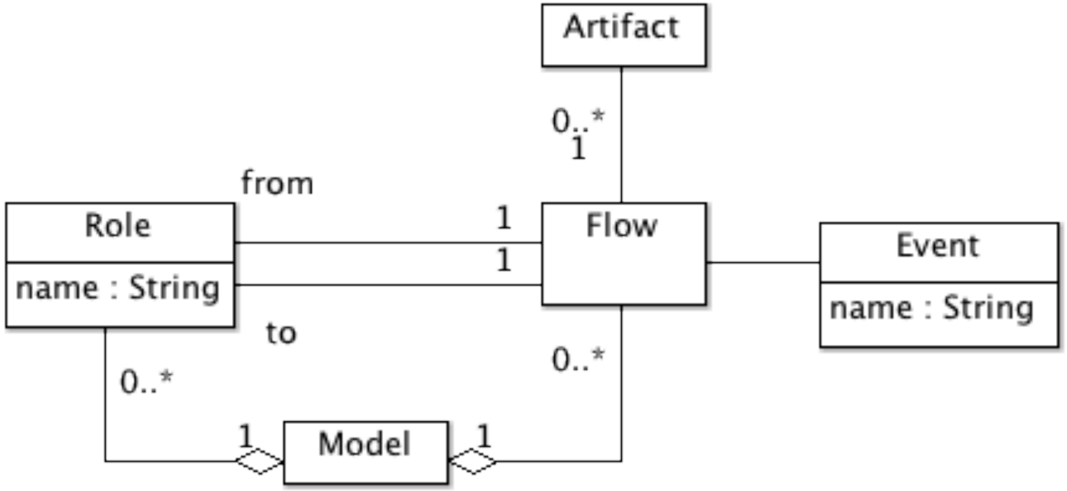
Making UCD more SE oriented

A DSL for Contextual Design

- Abstract Syntax
  - The cornerstone of a language definition
  - We define an abstract syntax for the main models in the CD modelling language
    - Flow models
    - Artifact models (equivalent to class models in UML so not considered further)
    - Cultural models
    - Sequence models

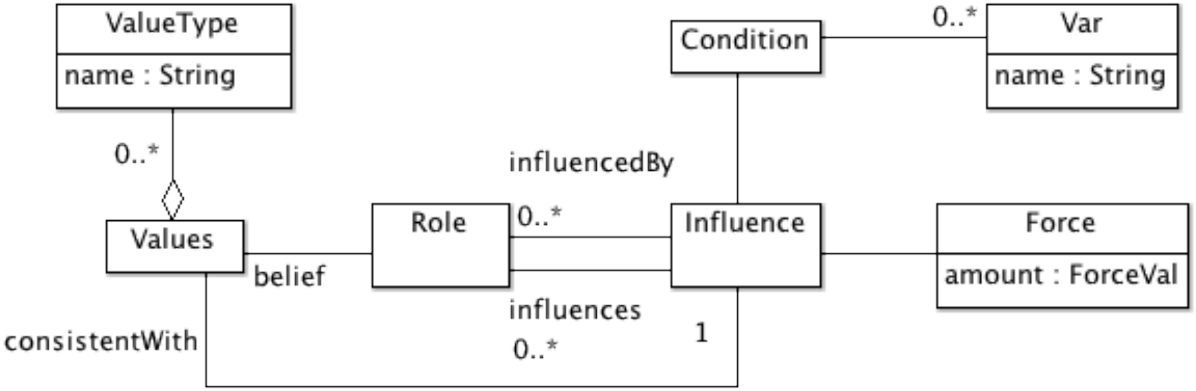
Abstract Syntax: Flow Model

- Model is the top-level container
- A Model consists of a collection of roles with flows between them
- Each flow represents an interaction between roles and is labelled with the event generated by it, the artifacts involved.
- Well-formedness: *every role must have a unique name*



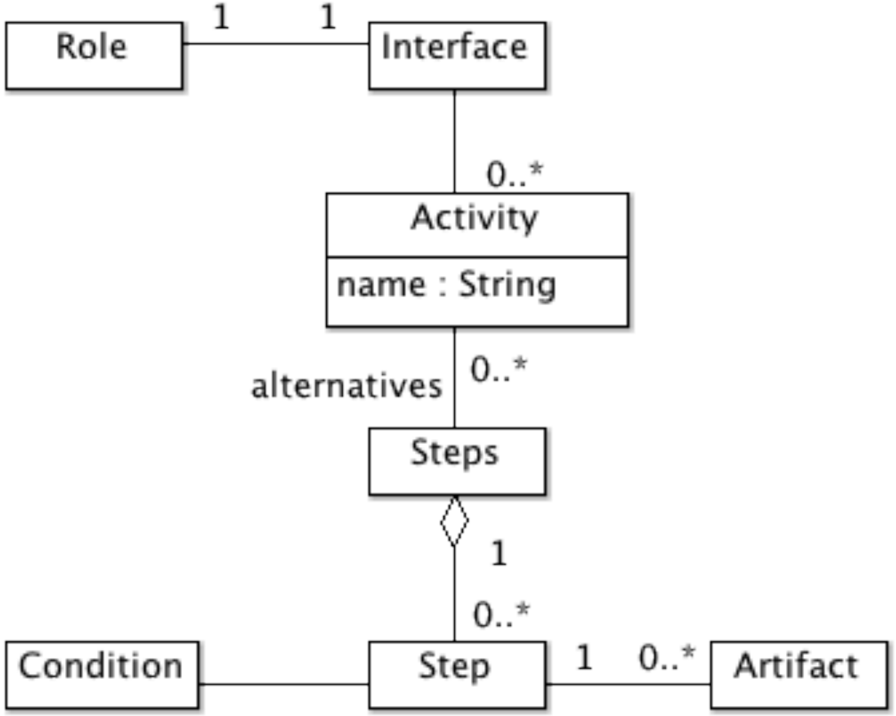
Abstract Syntax: Cultural Model

- Each **Influence** has a **Force** associated with it (weak to strong)
- Each **role** manages a collection of personal beliefs (**Values**)
- An Influence together with its Force defines a condition which must be met by any valid instance of Values associated with an influenced Role.
- Well-formedness: *Influence: the set of variable names in the condition must be a subset of the value type names associated with the belief values of an influenced role.*



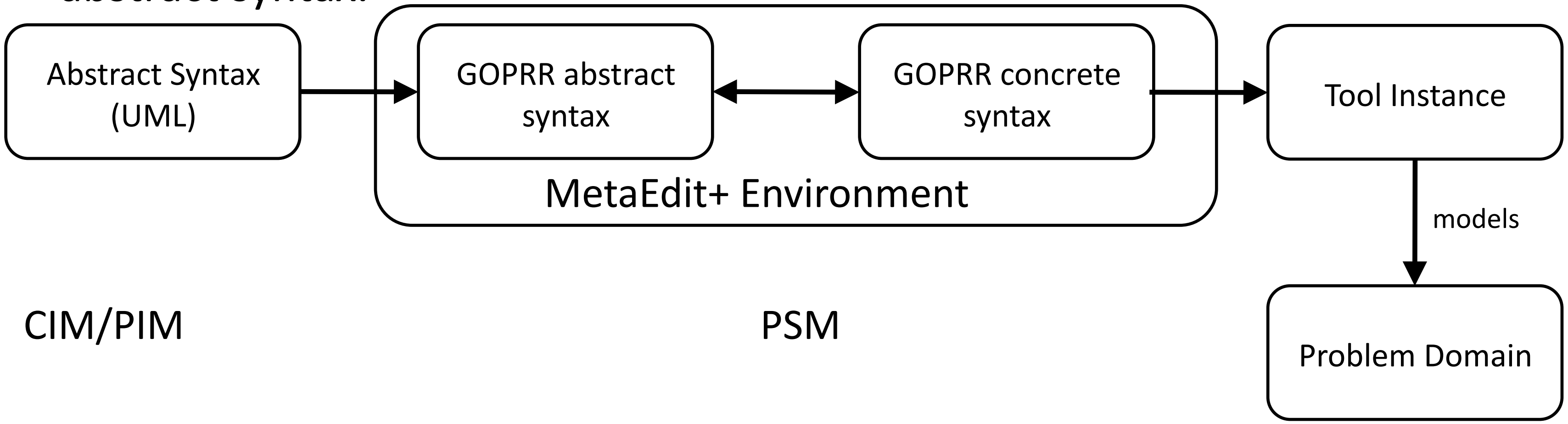
Abstract Syntax: Sequence Model

- Each Role has an Interface of Activities.
- Each Activity has a number of alternative step assemblies (Steps) that reflect the options that an individual performs in response to an event.
- Each individual step processes artifacts and must satisfy a collection of belief values.
- The idea is that a step cannot be performed unless it is consistent with the beliefs of an individual.

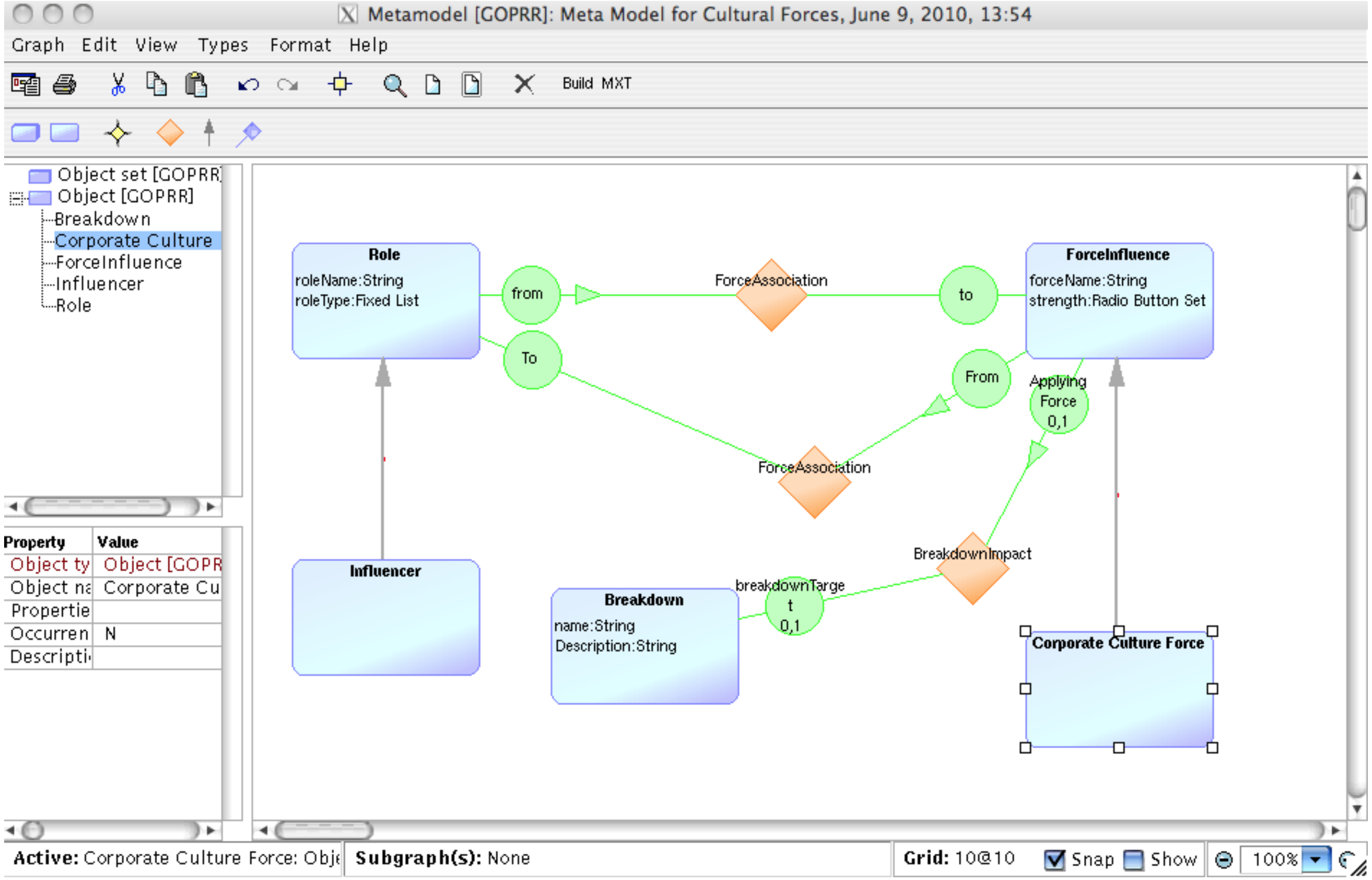


Concrete Syntax

- The complete abstract syntax for CD is large so we focus on the Cultural Forces model as it addresses areas of the systems design process not normally addressed.
- Translation into GOPRR meta modelling syntax for MetaEdit+.
- The tool supports the creation of a concrete syntax – the notations and graphical elements and their binding to the GOPRR equivalent of the abstract syntax.



Concrete Syntax

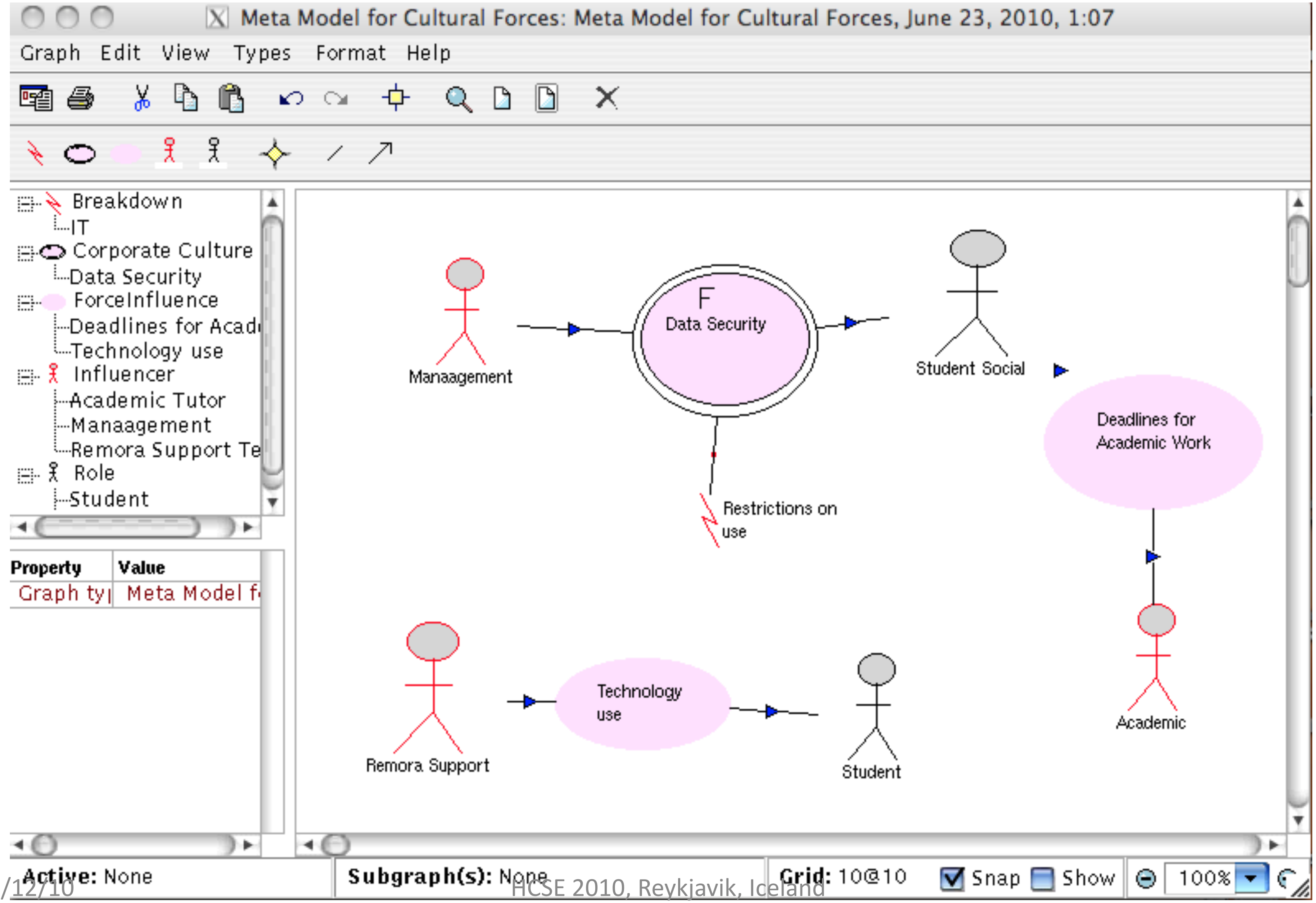


Metamodel [GOPRR]: Meta Model for Cultural Forces, June 9, 2010, 13:54

Property	Value
Object ty	Object [GOPR
Object na	Corporate Cu
Propertie	
Occurren	N
Descripti	

Active: Corporate Culture Force: Obje **Subgraph(s):** None **Grid:** 10@10 ☑ Snap □ Show 100%

Modeling Cultural Forces



Meta Model for Cultural Forces: Meta Model for Cultural Forces, June 23, 2010, 1:07

- Breakdown
  - IT
- Corporate Culture
  - Data Security
- ForceInfluence
  - Deadlines for Acad
  - Technology use
- Influencer
  - Academic Tutor
  - Manaagement
  - Remora Support Te
- Role
  - Student

Property	Value
Graph ty	Meta Model f

Active: None **Subgraph(s):** None **Grid:** 10@10 ☑ Snap □ Show 100%

Concluding remarks

- Our motivating example illustrated the problem that arises when core artifacts from the UCD process do not readily translate to the software engineering community
- Need to converge on a science of design
  - How can outputs from UCD be modeled so that they can be integrated with SE practice
- CD appears to be useful bridging methodology
  - But CD has an informal semantics – this limits tooling opportunities
- We have described CD can be given a formal syntax and we have outlined semantics for the method
- Issues of evaluation – will UCD experts use such tools?