# Derivation of a Dialog Model from a Task Model by Activity Chain Extraction

Kris Luyten[†] Tim Clerckx[†]
Karin Coninx[†] Jean Vanderdonckt[††]

[†] Limburgs Universitair Centrum
Expertisecentrum Digitale Media
[††]Université Catholique de Louvain
Institut d'Administration et de Gestion

DSV-IS 2003, 4–6 June, Madeira, Funchal

# 1. Overview

- Introduction
- Dygimes
- The Task Specification
- Dygimes Design Cycle
- The Dialog Specification
- Simple Case
- Future Work
- Questions

# 2. Introduction: Our Goals

- Decrease the gap between the Task specification and the Concrete User Interface (presentation)
  - Companies designers were interested in Task Modelling
  - Companies developers were interested in the Presentation
  - They did not want to synchronise the different models manually
- Direct prototyping
- Integrate it in a framework for runtime support

# 3. Dygimes

- Research framework
- Dygimes stands for: "**DY**namically **G**enerating **I**nterfaces for **M**obile and **E**mbedded **S**ystems"
- Dygimes is "yet another" tool using models for the creation of multi-device UIs
- it provides (historically ordered):
  - Multi-platform UIs given an abstract XML description
  - Integration with Component Based Software Development
  - Automatic layout management & presentation styles
  - Support for CTT Task Models

# 4. The Task Specification

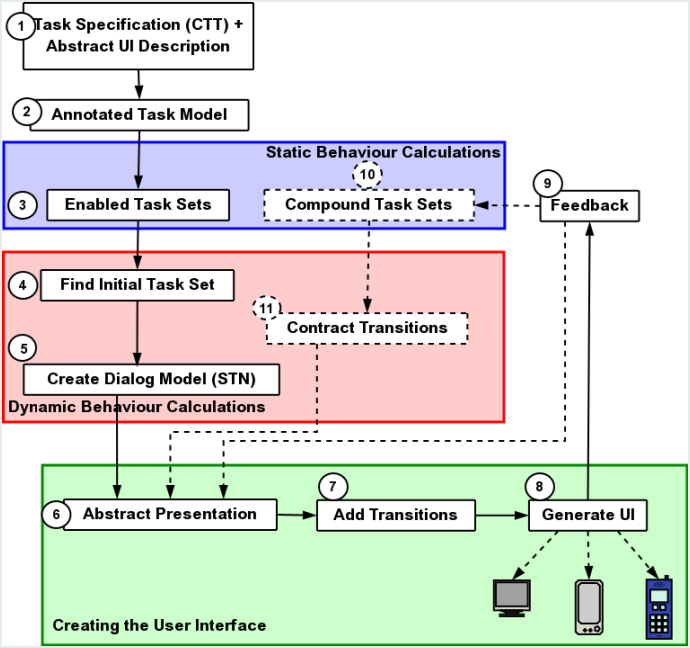
- We use the ConcurTaskTree notation:
  - Hierarchical structure
  - Temporal operators
  - Graphical syntax and tool support
- Essential in our system: *Enabled Task Sets*

  A set of tasks that are logically enabled to start their performance during the same period of time. [Paternò, 2000]

# 5. Dygimes design cycle

In short:

- Create a Task Model with ConcurTaskTrees
- Add abstract UI descriptions to the tree
- Use this as an input for Dygimes

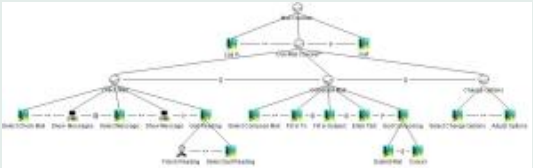


# 6. The Dialog Specification: A State Transition Network (STN)

- The Enabled Task Sets are calculated
- Each ETS is a node in the STN
  - Identify initial ETS
- Each node is a user interface window
- Transitions between nodes are identified by inspecting the temporal relations in the CTT
  - A transition is a task
  - Identify all "candidate" transitions
  - Detect transition tasks between ETS
- Identify transitions which lead to an *accept* state

# 7. A simple Case: An Email Application

- Simple task specification for checking/sending email
- Annotated with abstract UI descriptions



# 8. The disadvantages of the approach

- *Empirical*: can not prove correctness of dialog specification wrt task specification
- Difficult to add *new kinds of operators*: the set of rules has to be extended
- *Complexity* implementation is exponential wrt the set of operators that have to be processed

# 9. Future Work

- Use a formal notation (e.g. Kripke semantics, prove correctness)
- Less complex, more generic rule set for processing
- Add immediate designer feedback
- Integrate easy scenario testing
- Refactor for multi/multiple/very small devices

# 10. Questions

Questions, remarks, suggestions, . . .