

UNIL | Université de Lausanne

**Information Systems Institute**

---

Introduction

---

**REQUIREMENTS ANALYSIS**

- scenario-based design
- requirements analysis
- **task modeling**

---

DESIGN

- activity design
- information design
- interaction design

---

USABILITY EVALUATION

- prototyping
- usability testing
- documentation

---

Science of design

---

Analyse, modélisation et conception

## Design of Interactive Software

### 3. Task and goal modeling

HEC Lausanne - 1015 Lausanne - Switzerland - Tel. +41 21 692.3416 - yves.pigneur@unil.ch - http://www.hec.unil.ch/yp

Université de Lausanne

---

## Agenda

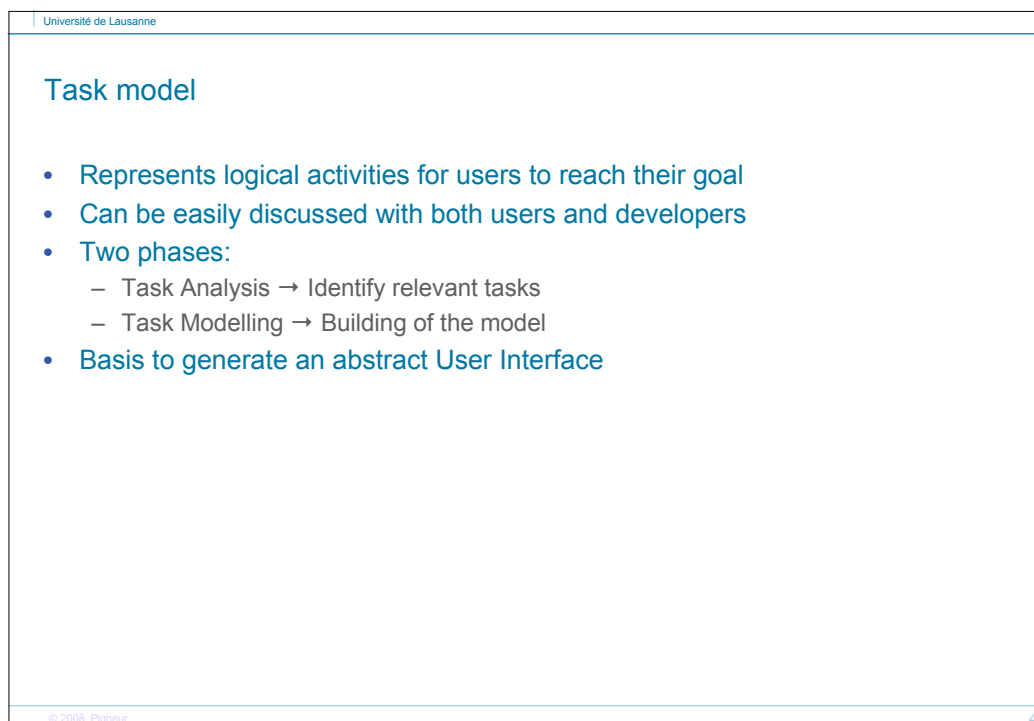
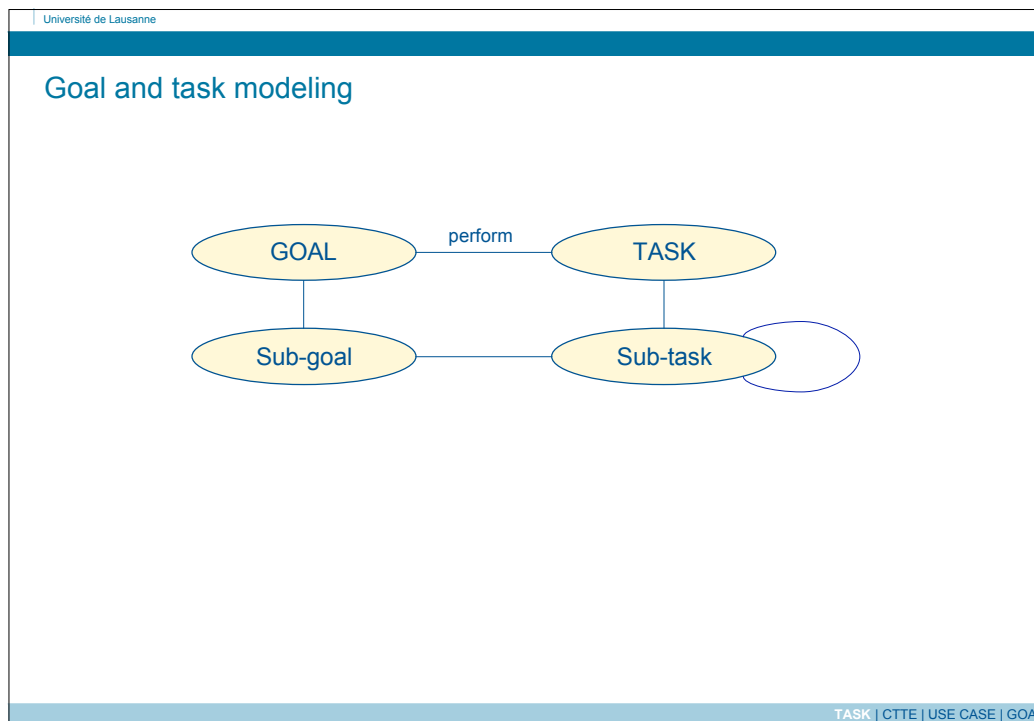
- **TASK MODEL DIAGRAM**
  - Hierarchical Task Analysis (HTA)
  - GOMS
  - UAN
- **ConcurTaskTree (& CTTE tool)**

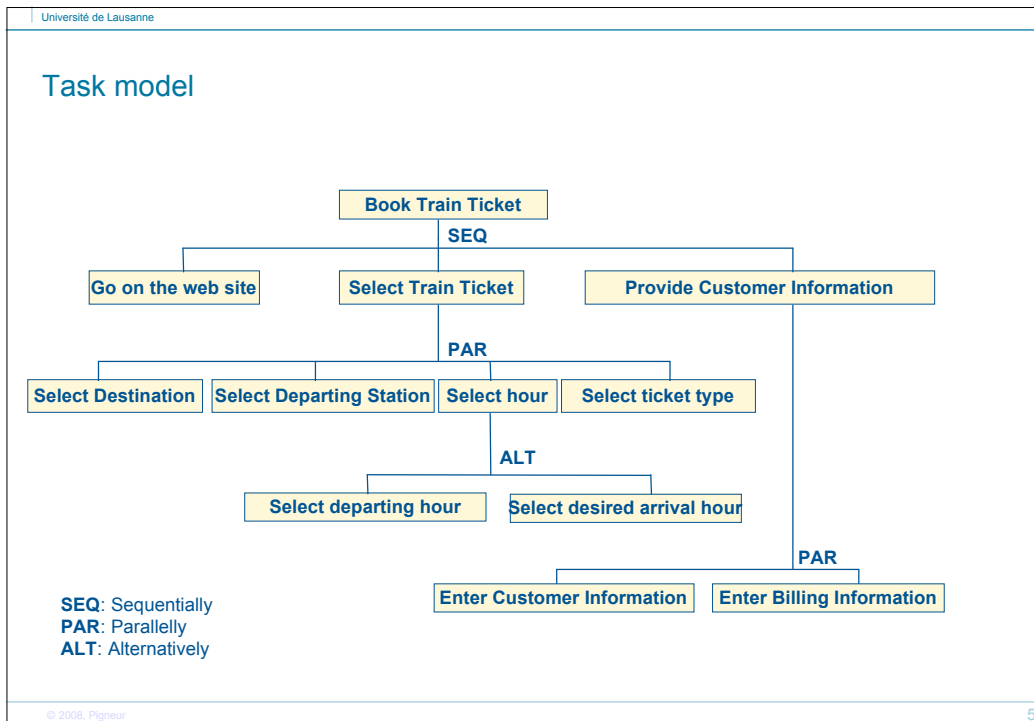
Mori, G., Paterno F., Santoro C. (2002) "CTTE: Support for Developing and Analyzing Task Models for Interactive System Design" *IEEE Transactions on Software Engineering*, 28 (9) pp.797-813
- **USE CASE**

For your information:

- **GOAL-BASED requirement engineering**
  - *Kaos*
  - *i\**
  - *Crews-L'écritoire*

TASK | CTTE | USE CASE | GOAL





Université de Lausanne

## VESALE > Task analysis

IHM > Analyse de la tâche

### Structure du chapitre

Next ▶

- ▼ Structure du chapitre
  - ▶ Objectifs de l'analyse de la tâche
  - ▶ Notion de tâche
  - ▶ Objets et buts de l'analyse
  - ▶ Les entrées nécessaires
  - ▶ Les résultats attendus
  - ▶ Dérivation des spécif. ergo. de l'IHM
  - ▶ Dérivation des spécif. fonctionnelles
  - ▶ Bibliographie

Le chapitre est structuré de la manière suivante :

- Objectifs de l'analyse de la tâche
  - Perspectives de l'analyse
- Notion de tâche
- Aperçu global de l'analyse de la tâche
- Inputs de l'analyse de la tâche
  - Exemple de fonctionnement réel d'un logiciel GPS
- Outputs de l'analyse de la tâche
  - Stéréotype des utilisateurs
  - Contexte de travail
  - Critères d'utilité et utilisabilité
  - Paramètres descriptifs de la tâche
  - Structuration de la tâche
- Techniques d'analyse
- Choix du style d'interaction et des attributs du dialogue
  - Dérivation du SDI en fonction du stéréotype d'utilisateur
  - Dérivation du SDI en fonction des paramètres descriptifs de la tâche interactive
  - Dérivation du SDI en fonction de la complexité du traitement
  - Dérivation des attributs du dialogue
    - Dérivation des attributs du dialogue : tableau
    - Sélection de SDI : exemple
- Spécifications fonctionnelles
- Graphe d'enchaînement des fonctions
- Bibliographie

Page 1 / 25 Next ▶

<http://www.hec.unil.ch/fbodart/IHM/Chap4/structure.html>

© 2008, Pigneur 6

Université de Lausanne

### VESALE > Task analysis > 3 perspectives

The diagram illustrates three perspectives on a task: Performer perspective, Organizational perspective, and Workstation perspective, all centered around a 'Task'. A vertical flow shows the task's progression: Abstract task, Planned task, and implemented task.

<http://www.hec.unil.ch/fbodart/IHM/Chap4/structure.html>

© 2008, Pigneur

7

Université de Lausanne

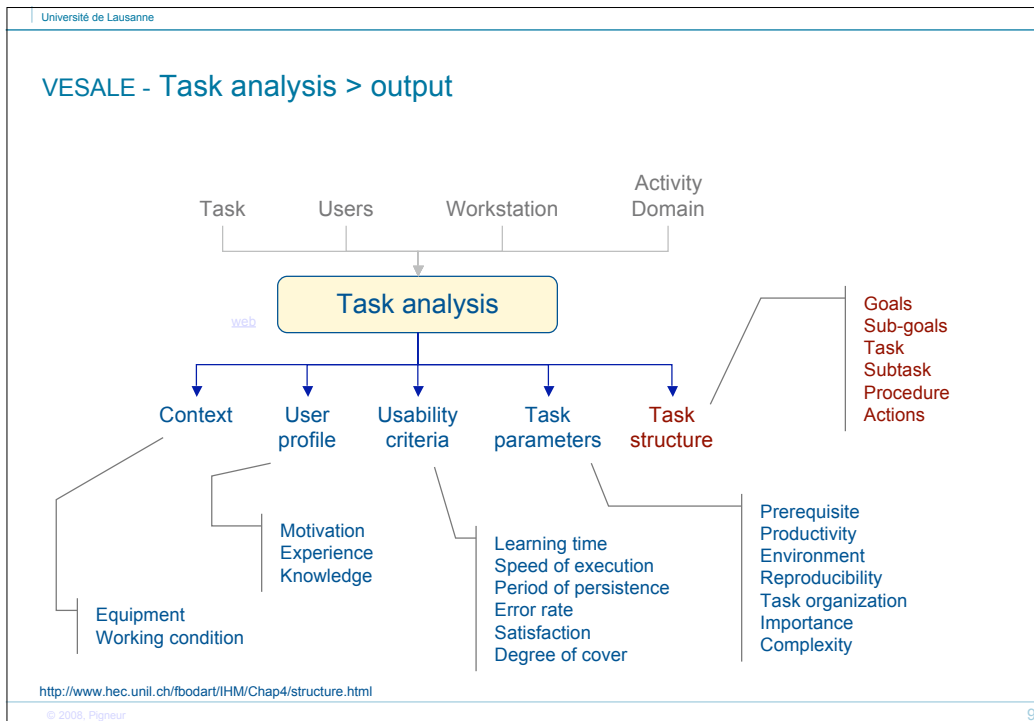
### VESALE - Task analysis > input

The diagram shows 'Task analysis' as a central process receiving input from 'Task', 'Users', 'Workstation', and 'Activity Domain'. A list of methods includes Direct observation, Indirect observation, Oral interview, Written questionnaire, Concurrent protocol, Retrospective protocol, and Use case. An image of a navigation screen is also present.

<http://www.hec.unil.ch/fbodart/IHM/Chap4/structure.html>

© 2008, Pigneur

8



- Université de Lausanne
- ### Uses of task models
- Improve understanding of applications
  - Record results of multi-disciplinary discussions and meetings
1. Requirement analysis
  2. Design of interactive applications
  3. Usability evaluation
- Context-based help for users
  - Document content and structure
- [Palanque, 2000] [Paterno, 2002]
- © 2008, Pigneur
- 10

Université de Lausanne

## Software engineering and human-computer interaction

- Availability of **flexible and expressive notations**
  - Design language > “Bringing design to software” [Rheinfrank and Evenson, 1996]
- Need for **systematic methods**
  - Analysis and uses of task models
- Support of the **reuse** of good design solutions
  - to problems which occur across many applications
- Availability of **automatic tools**
  - To support the phase of the design cycle


[Palanque, 2000] [Paterno, 2002]

© 2006, Pigneur 11

Université de Lausanne

## *Hierarchical Task Analysis (HTA)*

- breaking down the steps of a task (process) performed by a user, viewed at different levels of detail.
- Each step can be decomposed into lower-level sub-steps, thus forming a hierarchy of sub-tasks.
- The highest level of detail might be something like:
  - open the word processor -> type your document -> print it -> quit.
- However, opening a word processor is not a one-step process.
- It might break down into something like:
  - locate the word processing application icon -> click on the icon -> select Open from the File menu.



© 2006, Pigneur 12

Université de Lausanne

## Goals, Operators, Methods, and Selection (GOMS)

- **Goals**
  - Goals are what the user is trying to accomplish.
  - These can be defined at various levels of abstraction
  - Higher-level goals are decomposable into sub-goals, and are arranged hierarchically.
- **Operators**
  - Operators are the elementary perceptual, motor or cognitive actions that are used to accomplish the goals
  - Operators are not decomposable: they are atomic elements in the GOMS model.
- **Methods**
  - Methods are the procedures that describe the sequence of sub-goals and operators necessary to achieve the desired goal.
- **Selection rules**
  - Selection rules specify which method should be used to satisfy a given goal, based on the context.
  - Since there may be several different ways of achieving the same goal, selection rules represent the user's knowledge of which method must be applied to achieve the desired goal
  - Selection rules generally take the form of a conditional statement

[Paternò, 2000] [Kjorstein](#)

© 2008, Pigneur 13

Université de Lausanne

## Goals, Operators, Methods, and Selection (GOMS) Illustration

Goal: Save the document

Method 1: Use Save Menu

1. Move mouse to File Menu
2. Left-click mouse
3. Move mouse to save option
4. Left-click mouse

Method 2: Use Save Shortcut

1. Hit CONTROL-S on the keyboard

For user George:  
if on computer X, use the "Use Save Menu" method,  
otherwise use the "Use Save Shortcut" method.





[Paternò, 2000] [Kieras](#)

© 2008, Pigneur 14

Université de Lausanne

## ConcurTaskTree (CTT) concepts

- Focus on activities
- Hierarchical structure
  - Decomposition in sub-tasks
  - Inheritance (of temporal constraints for example)
- Graphical syntax
- Concurrent notation
- Task allocation
  - User task, application task, interaction task, abstract task

- Objects
  - User interface object
  - Application domain objects

[Paternò, 2002]

TASK | CTTE | USE CASE | GOAL

Université de Lausanne

## ConcurTaskTree (CTT) temporal operators

>>> LOTOS (Language Of Temporal Ordering Specifications)

$T1 \parallel T2$	Independent concurrency
$T1 \square T2$	Choice
$T1 \parallel\!\!\!  T2$	Concurrency with information exchange
$T1 \mid\!\!\!  T2$	Order independence
$T1 \lbracket T2$	Deactivation
$T1 \gg T2$	Enabling
$T1 \parallel\!\!\!  \gg T2$	Enabling with information passing
$T1 \lbracket T2$	Suspend-resume
$T1^*$	Iteration
$T1 (n)$	Finite iteration
$[T1]$	Optional task
$T1$	Recursion

LOTOS specification

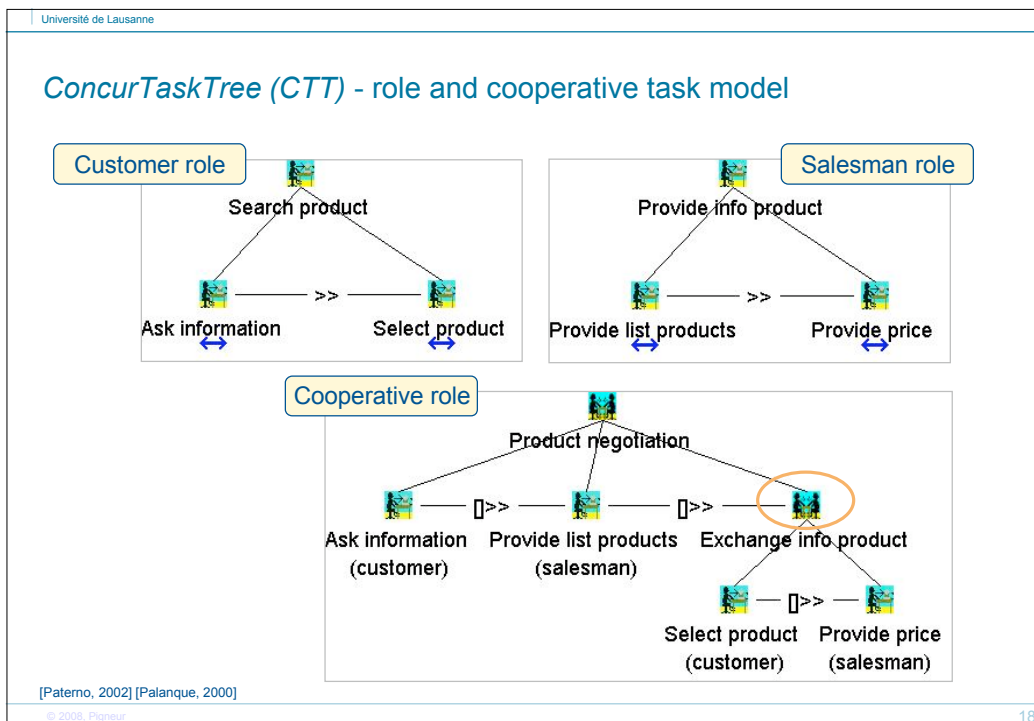
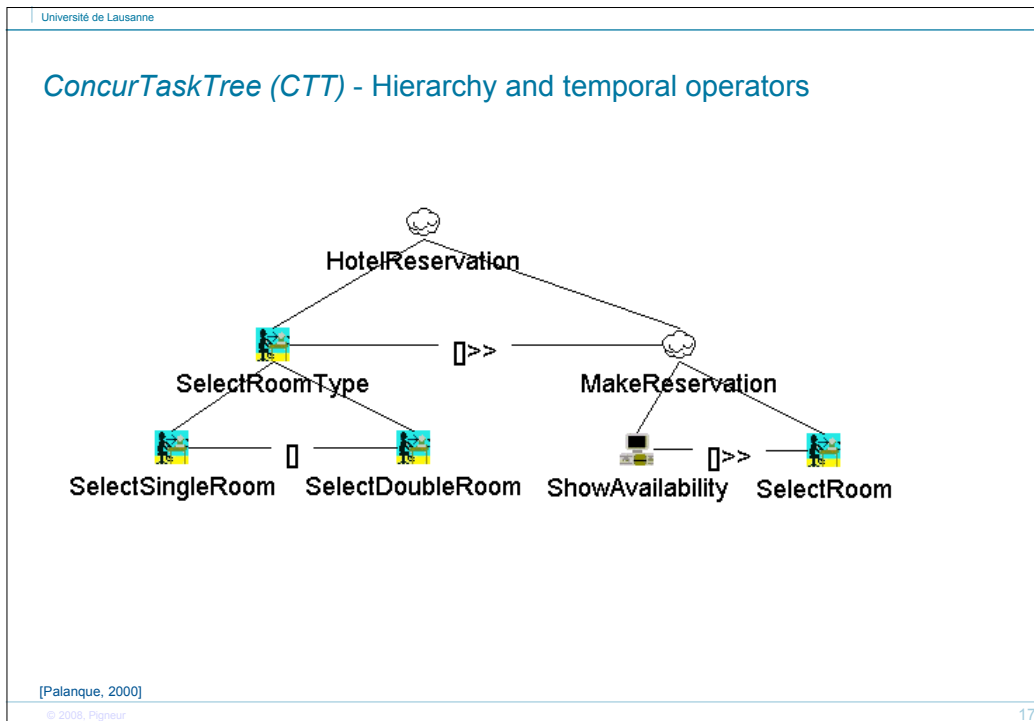
```

specification
  EnableAccess_spec[InsertCard, RequirePassword, InsertPassword]:
  library
    BOOLEAN,
    NATURAL,
    endlib
  behaviour
    InsertCard_proc[InsertCard]
    >> RequirePassword_proc[RequirePassword]
    >> InsertPassword_proc[InsertPassword]
  where
    process InsertCard_proc[InsertCard]:exit=
      InsertCard; exit
    endproc
    process RequirePassword_proc[RequirePassword]:exit=
      RequirePassword; exit
    endproc
    process InsertPassword_proc[InsertPassword]:exit=
      InsertPassword; exit
    endproc
  endspec
        
```

Save LOTOS specification | Close

[Paternò, 2002]

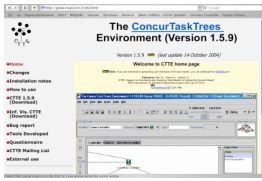
16



Université de Lausanne

## ConcurTaskTree environment (CTTE)

- Editing of task models
- Multiple interactive views of the specification
- Editing task models for cooperative applications
- Using informal descriptions in supporting modelling
- Checking completeness of the specification
- Saving the specification in various format
- Comparing task models
- Automatic expansion of task patterns
- SIMULATION



The screenshot shows the ConcurTaskTree Environment (Version 1.5.9) interface. It features a menu bar with options like File, Edit, View, Info, Insert, Tools, and Help. Below the menu is a toolbar with various icons for editing and viewing. The main workspace displays a hierarchical task tree with nodes such as 'AccessATM', 'EnableAccess', 'Access', and 'CloseAccess'. An 'Overview' window on the right provides a smaller view of the entire tree structure.

[Paternò, 2002]

© 2008, Pigneur

19

Université de Lausanne

## CTTE > editing a task model

File Edit View Info Insert Tools Help

Global view Local view Dialog 16

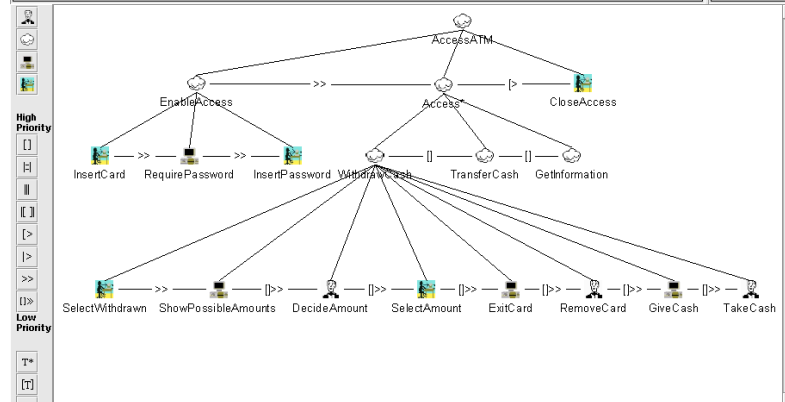
10% 84% 200%

Current Task: Identifier: RemoveCard Category: User Type: Frequency:

Platform:  Pda  Desktop  Cell  Vocal

High Priority

Low Priority



The diagram shows a hierarchical task tree for the 'RemoveCard' task. The root node is 'AccessATM', which branches into 'EnableAccess', 'Access', and 'CloseAccess'. 'EnableAccess' leads to 'InsertCard' and 'RequirePassword'. 'Access' leads to 'InsertPassword', 'WithdrawCash', 'TransferCash', and 'GetInformation'. 'CloseAccess' leads to 'SelectWithdraw', 'ShowPossibleAmounts', 'DecideAmount', 'SelectAmount', 'ExitCard', 'RemoveCard', 'GiveCash', and 'TakeCash'. The 'RemoveCard' node is highlighted, indicating it is the current task being edited.

Overview

<http://giove.cnuce.cnr.it/ctte.html>

© 2008, Pigneur

20

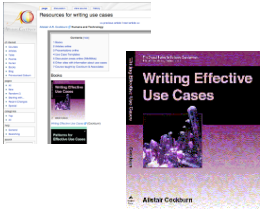
2\_task.ppt v1.3 (February 2008) /yp

10

Université de Lausanne

## Use case (I)

- “A use case captures a contract between the stakeholders of a system about its behaviour.
- Different sequences of behaviour, or scenarios, can unfold, depending on the particular requests made and conditions surrounding the requests. The use case collects together those different scenarios.
- Use cases are fundamentally a text form, although they can be written using flow charts, sequence charts, Petri nets, or programming languages.
- Under normal circumstances, they serve to communicate from one person to another, often to people with no special training.
- Simple text is, therefore, usually the best choice.



[Cockburn, 2000] <http://alistair.cockburn.us/usecases/usecases.html>

TASK | CTTE | USE CASE | GOAL

Université de Lausanne

## Use case > example (1)

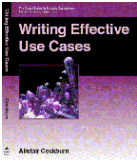
### USE CASE 1: BUY STOCKS OVER THE WEB

**Primary Actor:** Purchaser  
**Scope:** Personal Advisors / Finance package ("PAF")  
**Level:** User goal  
**Stakeholders and Interests:**  
 Purchaser - wants to buy stocks, get them added to the PAF portfolio automatically.  
 Stock agency - wants full purchase information.  
**Precondition:** User already has PAF open.  
**Minimal guarantee:** sufficient logging information that PAF can detect that something went wrong and can ask the user to provide details.  
**Success guarantee:** remote web site has acknowledged the purchase, the logs and the user's portfolio are updated.  
**Main success scenario:**

1. User selects to buy stocks over the web.
2. PAF gets name of web site to use (E\*Trade, Schwabb, etc.) from user.
3. PAF opens web connection to the site, retaining control.
4. User browses and buys stock from the web site.
5. PAF intercepts responses from the web site, and updates the user's portfolio.
6. PAF shows the user the new portfolio standing.

**Extensions:**

- 2a. User wants a web site PAF does not support:
  - 2a1. System gets new suggestion from user, with option to cancel use case.
- 3a. Web failure of any sort during setup:
  - 3a1. System reports failure to user with advice, backs up to previous step.
  - 3a2. User either backs out of this use case, or tries again.
- 4a. Computer crashes or gets switched off during purchase transaction:
  - 4a1. (what do we do here?)
- 4b. Web site does not acknowledge purchase, but puts it on delay:
  - 4b1. PAF logs the delay, sets a timer to ask the user about the outcome.
  - 4b2. (see use case *Update questioned purchase*)
- 5a. Web site does not return the needed information from the purchase:
  - 5a1. PAF logs the lack of information, has the user *Update questioned purchase*.



[Cockburn, 2000]

© 2008, Pigneur


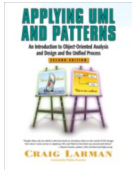
22

Université de Lausanne

## Definition

- “Users have goals (needs) and want computer systems to help meet them
- There are several ways to capture goals and system requirements
- Use cases are a mechanism to help keep it simple and understandable for all stakeholders
- They are stories of using a system to meet goals
  - **Use case:** A customer arrives at a checkout with items to purchase. The cashier uses the POS system to record each purchased item. The system presents a running total and line-item details. The customer enters payment information, which the system validates and records. The system updates inventory. The customer receives a receipt from the system and then leaves with the items.

[Jacobson, 1986]

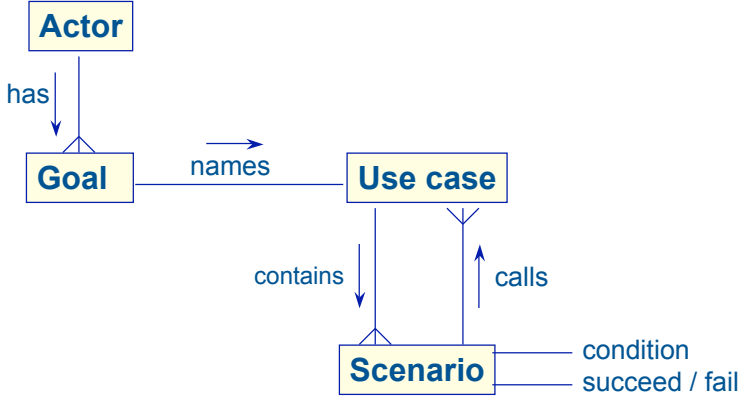



[Larman, 2004] [Pols, 2000] > CHAPTER 6

© 2008, Pigneur 23

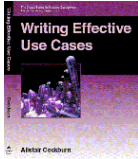
Université de Lausanne

## Use case, actor, goal, and scenario



```

    graph TD
      Actor[Actor] -- has --> Goal[Goal]
      Goal -- names --> UseCase[Use case]
      UseCase -- contains --> Scenario[Scenario]
      Scenario -- calls --> UseCase
      Scenario --- Condition[condition succeed / fail]
    
```



[Cockburn, 2000] <http://alistair.cockburn.us/usecases/usecases.html>

© 2008, Pigneur 24

## Use case, actor, goal, and scenario

- **An actor** is something with behaviour
- A **scenario** is a specific sequence of actions and interactions between actors and the system under discussion; it is also called a use case instance. It is one particular story of using a system, or one path through the use case
- A **use case** is a collection of related success and failure scenarios that describe actors using a system to support a goal.
  - **Main Success Scenario:** A customer arrives at a checkout with items to return. The cashier uses the POS system to record each returned item ...
  - **Alternate Scenarios:**
    - If the credit authorization is reject, inform the customer and ask for an alternate payment method
    - If the item identifier is not found in the system, notify the Cashier and suggest manual entry of the identifier code
    - If the system detects failure to communicate with ...



[Larman, 2004] [Pols, 2000] > CHAPTER 6

© 2008, Pigneur

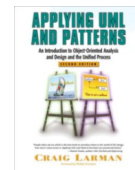
25

## Use case > definition (II)

- A set of use-case instances, where each instance is a sequence of actions a system performs that yields an observable result of value to a particular actor

A key attitude in use case work is to focus on the question  
 "How can using the system provide observable value to the user, or fulfil their goals?"  
 rather than merely thinking of system requirements in terms of a "laundry list" of features.

- Use cases are requirements
  - Define a promise or contract of how a system will behave
  - **Text documents, not diagrams**
    - Use case modelling = writing, not drawing
    - Even if UML defines a use case diagram to illustrate the names of use cases and actors, and their relationships



[Larman, 2004] [Pols, 2000] > CHAPTER 6

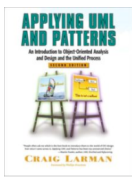
© 2008, Pigneur

26

Université de Lausanne

## Use case > types and formats

- **Black-box use cases**
  - User view (what? why?)
  - No internal workings of the system (how?)
  - SYSTEM RESPONSABILITIES
- **Brief**
  - Terse one paragraph summary
- **Casual**
  - Informal paragraph format
- **Fully dressed**
  - All steps and variations in detail
  - And supporting sections ...



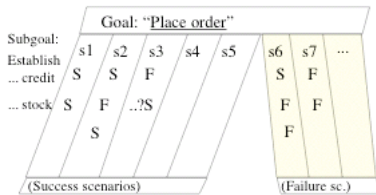
[Larman, 2004] [Pols, 2000] > CHAPTER 6

© 2008, Pigneur 27

Université de Lausanne

## Use case > Method

1. Identify the actors and their goals
2. For each use case, Write the simple case: goal delivers
  - The main success scenario
  - RESULT: Readable description of system's function
3. Write failure conditions as extensions
  - Each step can fail
  - RESULT: List of alternate scenarios
4. For each failure condition, Follow the failure till it ends or rejoins
  - Recoverable extensions rejoin main course, non recoverable ones fail directly
  - RESULT: complete use case
5. Note the data variations



[Cockburn, 2000] <http://alistair.cockburn.us/usecases/usecases.html>

© 2008, Pigneur 28

Université de Lausanne

## Use case > sections > goals and scope

**Goal in context:** process sales

**Scope and level:** company and summary

- A statement of the goal in context
- What system is being considered black box under design
  - Summary, primary task, sub-function ...

[Larman, 2004] [Pols, 2000] > CHAPTER 6

© 2008, Pigneur 29

Université de Lausanne

## Use case > sections > actors

**Primary actor:** Cashier

**Stakeholders and interests:**

Cashier: *wants accurate, fast, entry, and no payment errors ...*

Salesperson: *wants sales commissions updated*

Customer: *wants purchase and fast service with minimal effort ...*

- The principal actor that calls upon system services to fulfil a goal ...
- The stakeholders are to be satisfied by the use case

[Larman, 2004] [Pols, 2000] > CHAPTER 6

© 2008, Pigneur 30

Université de Lausanne

## Use case > sections > pre- and post-conditions

**Preconditions:** Cashier is identified and authenticated

**Success guarantees:** Sale is saved.  
Tax is correctly calculated.  
Accounting and inventory are updated. Commissions recorded.  
Receipt is generated

- **Precondition**  
state what must always be true before beginning a scenario in the use case
  - Assumed to be true
- **Success guarantees (post-condition)**  
state what must be true on successful completion of the use case
  - success end condition
  - failure end condition

[Larman, 2004] [Pols, 2000] > CHAPTER 6

© 2008, Pigneur 31

Université de Lausanne

## Use case > sections > trigger and main success scenario

**Trigger :** Customer arrives at POS checkout with goods to purchase.

**Main success scenario :**

1. Cashier starts a new sale.
2. Cashier enters item identifier.
3. System records sale line item and presents item description, price, and running total.  
Price calculated from a set of price rules.

*Cashier repeats steps 2-3 until indicates done.*

4. System presents total with taxes calculated. ...

- **Trigger:** indicates the event that starts the scenario
- **Main success scenario: basic flow**
  - From trigger to goal delivery
  - Does not include any conditions or branching
    - Interaction between actors
    - Validation (by the system)
    - State change by the system

[Larman, 2004] [Pols, 2000] > CHAPTER 6

© 2008, Pigneur 32

Université de Lausanne

## Use case > sections > trigger and main success scenario (II)

**Main success scenario :**

1. Cashier starts a new sale.
2. Cashier enters item identifier

*Cashier repeats steps 2-3 until indicates done*

3. System records sale line item and presents item description, and running total.
4. System presents total with taxes calculated. ...

- Two-column variation

[Larman, 2004] [Pols, 2000] > CHAPTER 6

© 2008, Pigneur 33

Université de Lausanne

## Use case > sections > extension

**Extensions:**

3a. Invalid identifier:

1. System signals error and rejects entry.

3b. There are multiple of same item category and tracking unique item identity not important:

1. Cashier can enter item category identifier and the quantity.

3-6a: Customer asks Cashier to remove an item from the purchase:

1. Cashier enters item identifier for removal from sale.
2. System displays updated running total ...

- **Alternative flows**
  - Success or failure
  - Condition causing branching (3a & 3b)
  - Handling: Action or name of sub.use case (1. 2. ...)

[Larman, 2004] [Pols, 2000] > CHAPTER 6

© 2008, Pigneur 34

Université de Lausanne

## Use case > sections > variations

**Technology and data variations:**

- 3a. Item identifier entered by bar code laser scanner (if bar code is present) or keyboard.
- 3b. Item identifier may be any UPC, EAN, JAN, or SKU coding scheme.
- 7a. Credit account information entered by card reader or keyboard.
- 7b. Credit payment signature captured on paper receipt.

But within two years,  
we predict many customers will want digital signature capture.

- **Constraints**
  - Input/output technology
  - Standards

[Larman, 2004] [Pols, 2000] > CHAPTER 6

© 2008, Pigneur 35

Université de Lausanne

## Use case > sections > special requirements

**Special requirements:**

- Touch screen UI on a large flat panel monitor. Text must be visible from 1 meter.
- Credit authorization response within 30 seconds 90% of the time.
- Robust recovery when access to remote services such the inventory system is failing.
- Language internationalization on the text displayed.
- Pluggable business rules to be insertable at steps 3 and 7.
- ...

- **Non-functional requirements**
  - Performance, reliability, usability, design constraints ...

[Larman, 2004] [Pols, 2000] > CHAPTER 6

© 2008, Pigneur 36

Université de Lausanne

## Template (I)

<b>USE CASE #</b>	< the name is the goal as a short active verb phrase >	
<b>Goal in Context</b>	<a longer statement of the goal in context if needed >	
<b>Scope &amp; Level</b>	<what system is being considered black box under design > <one of : Summary, Primary Task, Subfunction >	
<b>Preconditions</b>	<what we expect is already the state of the world >	
<b>Success End Condition</b>	<the state of the world upon successful completion >	
<b>Failed End Condition</b>	<the state of the world if goal abandoned >	
<b>Primary, Secondary Actors</b>	<a role name or description for the primary actor >. <other systems relied upon to accomplish use case >	
<b>Trigger</b>	<the action upon the system that starts the use case >	
<b>DESCRIPTION</b>	<b>Step</b>	<b>Action</b>
	1	<put here the steps of the scenario from trigger to goal delivery, and any cleanup after >
	2	<... >
	3	
<b>EXTENSIONS</b>	<b>Step</b>	<b>Branching Action</b>
	1 a	<condition causing branching > : <action or name of sub.use case >
<b>SUB-VARIATIONS</b>		<b>Branching Action</b>
	1	<list of variations >

[Cockburn, 2000] <http://alistair.cockburn.us/usecases/usecases.html>

© 2008, Pigneur

37

Université de Lausanne

## Template (II)

<b>RELATED INFORMATION</b>	<Use case name >
<b>Priority:</b>	<how critical to your system / organization >
<b>Performance</b>	<the amount of time this use case should take >
<b>Frequency</b>	<how often it is expected to happen >
<b>Channels to actors</b>	<e.g. interactive, static files, database, timeouts >
<b>OPEN ISSUES</b>	<list of issues awaiting decision affecting this use case >
<b>Due Date</b>	<date or release needed >
<b>...any other management information...</b>	<...as needed >
<b>Superordinates</b>	<optional, name of use case(s) that includes this one >
<b>Subordinates</b>	<optional, depending on tools, links to sub.use cases >

[Cockburn, 2000] <http://alistair.cockburn.us/usecases/usecases.html>

© 2008, Pigneur

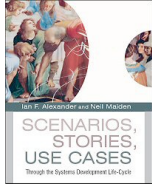
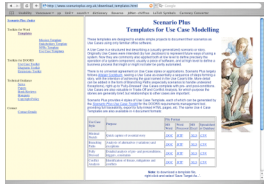
38

Université de Lausanne

## Other template

ID	Use Case Headings & Text	Actor	Use Case Level	Use Case Span	Use Case Scope	Review Status
2	UC-35 2.1.3 Fully-Defined Use Case		Surface	System Black-Box	In Scope	Draft
3	UC-36 2.1.3.1 Primary Scenario					
4	UC-37 Undefined Step					
5	UC-64 Undefined Step					
6	UC-63 Undefined Step					
7	UC-38 2.1.3.2 Alternative Paths					
8	UC-39 2.1.3.2.1 Unnamed Alternative Path					
9	UC-40 Undefined Step					
10	UC-66 Undefined Step					
11	UC-65 Undefined Step					
12	UC-41 2.1.3.3 Exceptions					
13	UC-42 2.1.3.3.1 Unnamed Exception					
14	UC-43 Undefined Step					
15	UC-68 Undefined Step					
16	UC-67 Undefined Step					
17	UC-44 2.1.3.4 Trigger					
18	UC-45					
19	UC-46 2.1.3.5 Preconditions					
20	UC-47					
21	UC-69					
22	UC-48 2.1.3.6 Stakeholders and Interests					
23	UC-49					
24	UC-71					
25	UC-70					
26	UC-50 2.1.3.7 Minimal Guarantees					
27	UC-51					
28	UC-72					
29	UC-52 2.1.3.8 Success Guarantees					
30	UC-53					
31	UC-73					
32	UC-54 2.1.3.9 Constraints					
33	UC-55					
34	UC-74					
35	UC-56 2.1.3.10 Non-Functional Requirements					
36	UC-57					
37	UC-76					
38	UC-75					
39						
40						
41						
42						
43						
44						
45						
46						
47						
48	All images and text © Scenario Plus 1997-2003					

Overview	Organisation Black-Box	Out of Scope	Draft
High	Organisation White-Box	In Scope	Proposed
Surface	System Black-Box		Accepted
Low	System White-Box		Rejected
Too Detailed	Component		On Hold
			Completed

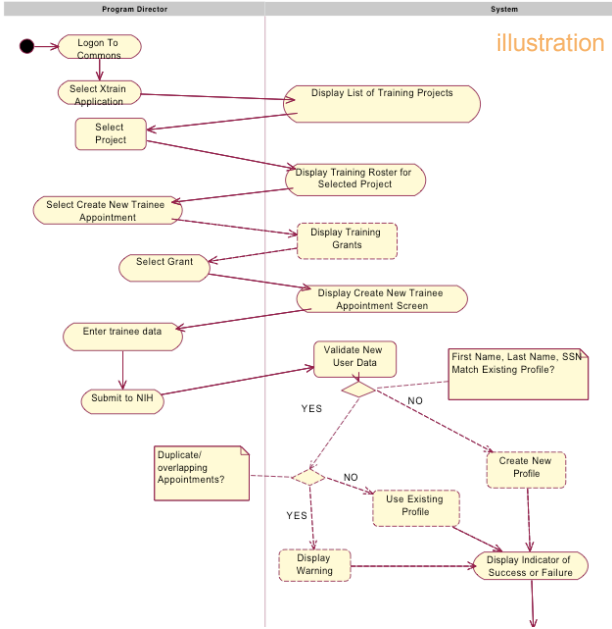
[Alexander, 2004] <http://www.scenarioplus.org.uk>

© 2008, Pigneur

Université de Lausanne

## UML diagram two-column variation

- See Interaction design



RNSolutions 2004

© 2008, Pigneur

Université de Lausanne

## Goal-based requirement engineering

- “RE is concerned with the identification of goals to be achieved by the envisioned system, the operationalization of such goals into services and constraints...”

The diagram illustrates the process of goal-based requirement engineering. It starts with 'domain knowledge' (represented by an open book) and 'goals' (represented by a cloud). Both lead to 'requirements, assumptions' (represented by a scroll). The arrow from 'goals' to 'requirements, assumptions' is labeled 'operationalization'. The scroll is labeled 'System Requirements'. Text labels include 'WHY?' near 'goals', 'WHAT?' near 'domain knowledge', and 'Mission statements & strategic objectives' near 'goals'.

[van Lamsweerde, 2000] [Rolland, 2003]

TASK | CTTE | USE CASE | GOAL

Université de Lausanne

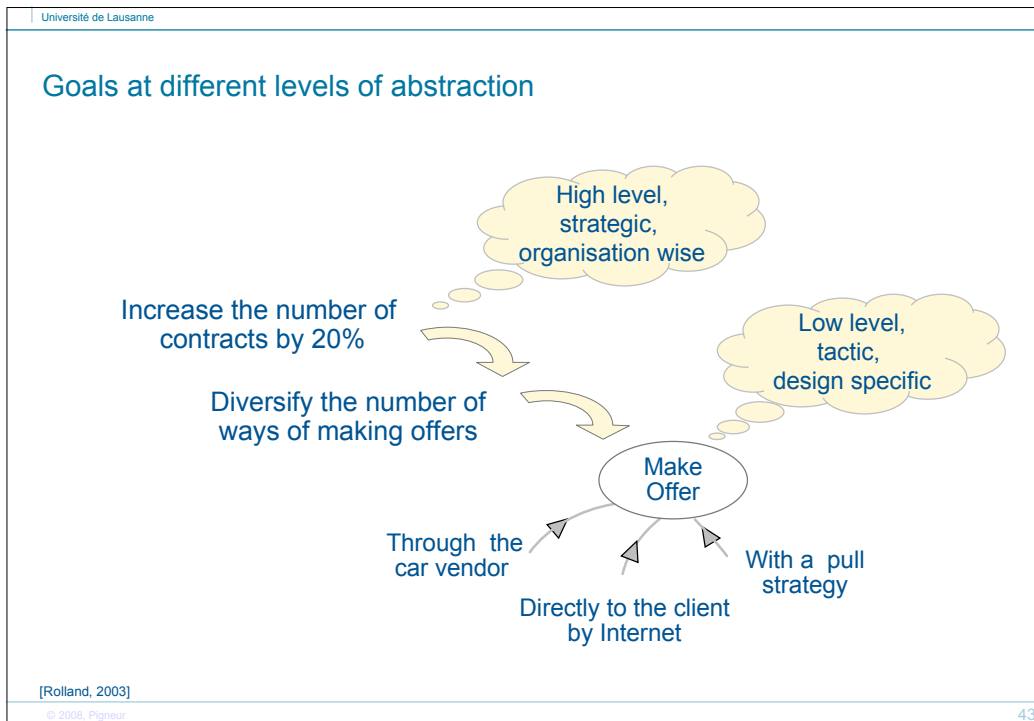
## Intentional models

- Encompass the world of things agents believe in, want, prove, ...
- Goals have been studied in AI, mostly as part of a formal framework for doing planning
- A goal is a desired state
  - Profits (year (2005))  $\geq$  \$1B (strategic goal)
  - Sales (VW beetle, week (24/03/2005))  $\geq$  5 (operational goal)

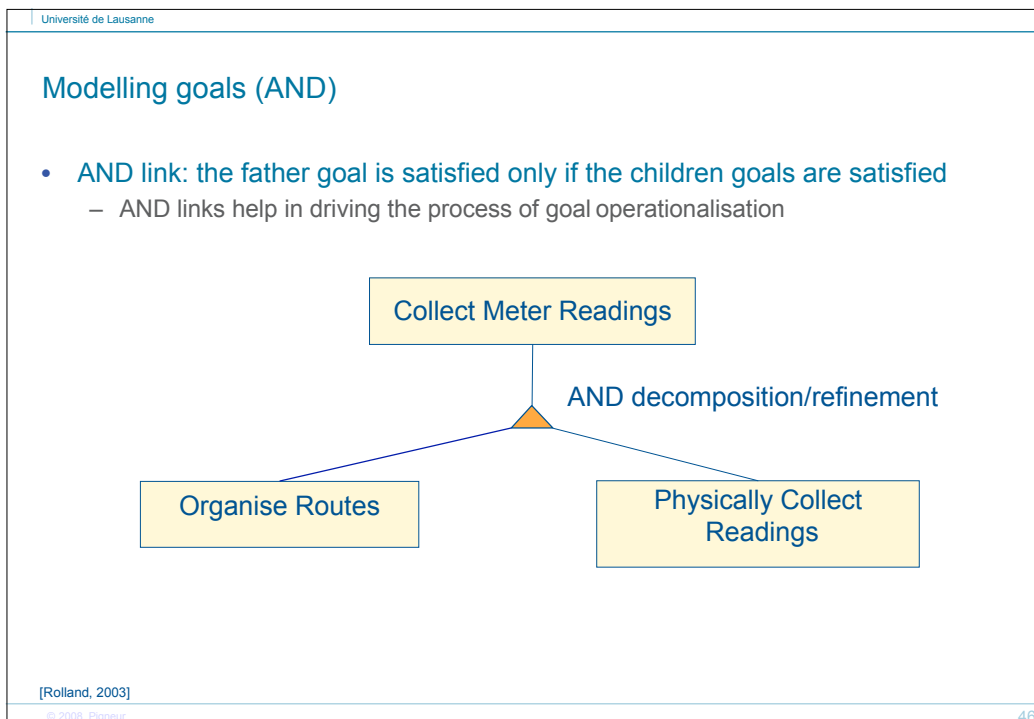
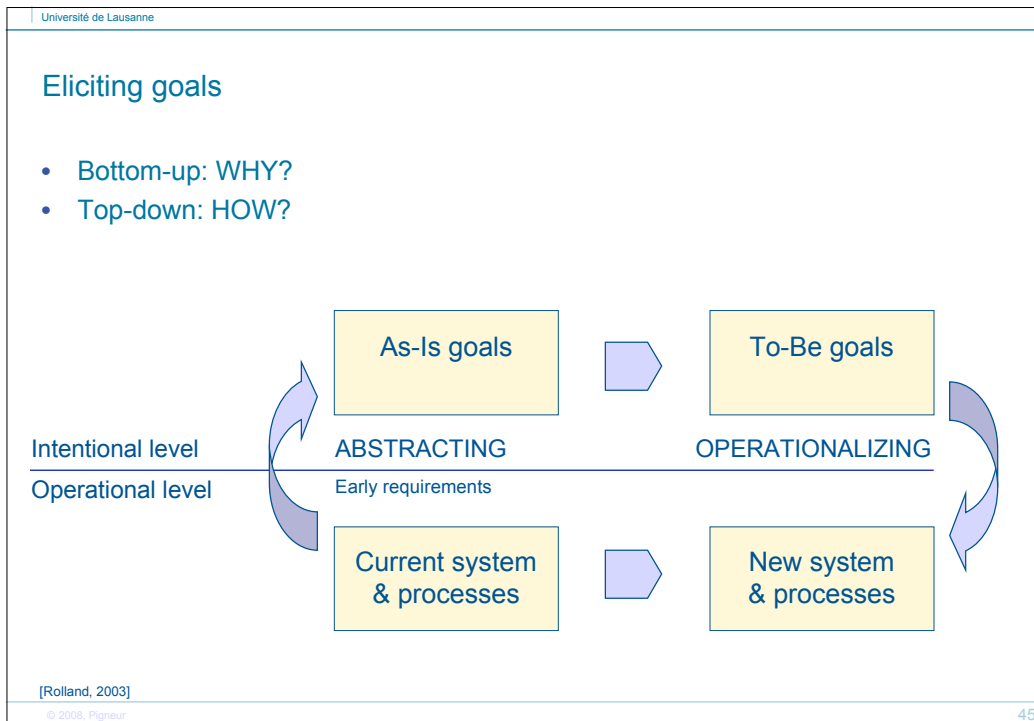
[Mylopoulos, 2004]

© 2008, Pigneur

42



- Université de Lausanne
- ### Goals at different types of concerns
- **functional goals**
    - what the system is expected to do
    - To be satisfied
  
  - **non-functional goals**
    - quality of the system behaviour
      - security, safety, accuracy, performance, cost, usability, adaptability
    - To be “satisfied”
      - Soft goal fulfilment is relative and “good enough”, rather than absolute and minimal
    - User-friendly [Interface]
- SOFT GOALS**
- [Rolland, 2003] [Mylopoulos, 2004]
- © 2008, Pigneur
- 44



Université de Lausanne

### Modelling goals (OR)

- OR link: the father goal is satisfied only if one of the children goals is satisfied

```

    graph TD
      A[Provide Customer support] -- OR refinement --> B[Provide 24-hours Telephone hot-line]
      A -- OR refinement --> C[Provide Internet forum]
    
```

[Rolland, 2003]  
© 2006, Pigneur

47

Université de Lausanne

### Goals links and dependencies

- + or - link: one goal contributes positively or negatively towards the fulfilment of another goal

```

    graph TD
      RT[Rapid transportation] -- + --> TP[Train progress]
      RT -- + --> ND[No delay]
      ST[Safe transportation] -- + --> NC[No collision]
      ND -- + --> TSB[Trains on same block]
      NC -- - --> TSB
    
```

[Rolland, 2003] [Mylopoulos, 2004]  
© 2006, Pigneur

48

Université de Lausanne

## Types of goals example

INTRODUCE	<i>Introduce a customer oriented culture</i>
IMPROVE	<i>Improve customer satisfaction for new installation request</i>
EXTEND	<i>Extend current liaison procedures</i>
CEASE	<i>Cease offering services to public</i>
MAINTAIN	<i>Maintain the safe and continuous provision of electricity</i>
ADAPT	<i>Automate the financial aspects for electricity supply</i>
REPLACE	<i>Replace manual meter reading by remote meter reading</i>

[Rolland, 2003]

© 2008, Pigneur 49

Université de Lausanne

## Types and formalization of goals

- **Generate behaviours**
  - **Achieve:**  $P \Rightarrow \diamond Q$       *Achieve [message sent in time]*
  - **Cease:**  $P \Rightarrow \diamond \neg Q$       *Cease [deliver electricity]*
- **Restrict behaviours**
  - **Maintain:**  $P \Rightarrow Q$       *Maintain [distance between train]*
  - **Avoid:**  $P \Rightarrow \neg Q$       *Avoid [train collision]*
- **Compare behaviours**
  - **Optimize:** maximise, minimise ...

Q: property will hold in all future states  
 $\diamond Q$ : property will hold in some future states

[Dardenne, 1993]

© 2008, Pigneur 50

Université de Lausanne

## Formalization of goals

example

Achieve [BookRequestSatisfied]

( $\forall$  bor: Borrower, b: Book, lib: Library)  
 Requesting (bor, b)  $\wedge$  b.subject  $\in$  lib.coverageArea

$\Rightarrow \diamond (\exists bc: BookCopy) (Copy(bc, b) \wedge Borrowing(bor, bc))$

Q: property will hold in all future states  
 $\diamond$ Q: property will hold in some future states

KAOS [van Lamsweerde, 2000] [Dardenne, 1993]

© 2008, Pigneur 51

Université de Lausanne

## Goals, stakeholders and conflicts

- **Goals may be owned by stakeholders**
  - "train frequency increased" (passengers)
  - "number of passengers increased" (train company)
- **Achieving goals require co operation among agents/actors**
  - DEPENDENCIES < actor1, goal g, actor2>
  - goal g requires the cooperation between actor1 (dependee) and actor2 (depender)
- **potential for conflicting viewpoints**
  - "card kept when KO-customer" (banks)
  - "card returned when KO-customer" (customers)

[Rolland, 2003]

© 2008, Pigneur 52

Université de Lausanne

## Social models: actors & dependencies

- **Social relationships among actors**
  1. Work action approach
    - speech act, ...
  2. **Social dependency approach** I\* (distributed intentionality)
    - **Goals, beliefs, commitments** and
    - inter-actor **dependencies**, intentional relationships among **actors**: one actor depends on another
      - to satisfy a goal,
      - execute a task, or
      - furnish a resource

```

graph LR
    TC((Train Controller)) --- D1(DoorsClosed WhileMoving) --- P((Passenger))
    TC --- D2(NoDelay) --- P
    TC --- D3(Alarm Notified) --- P
    TC --- D4(Accurate TrainPosition) --- P
    TC --- D4 --- TS((Tracking System))
  
```

[Rolland, 2003] [Mylopoulos, 2004] [Yu, 1995]

© 2008, Pigneur

53

Université de Lausanne

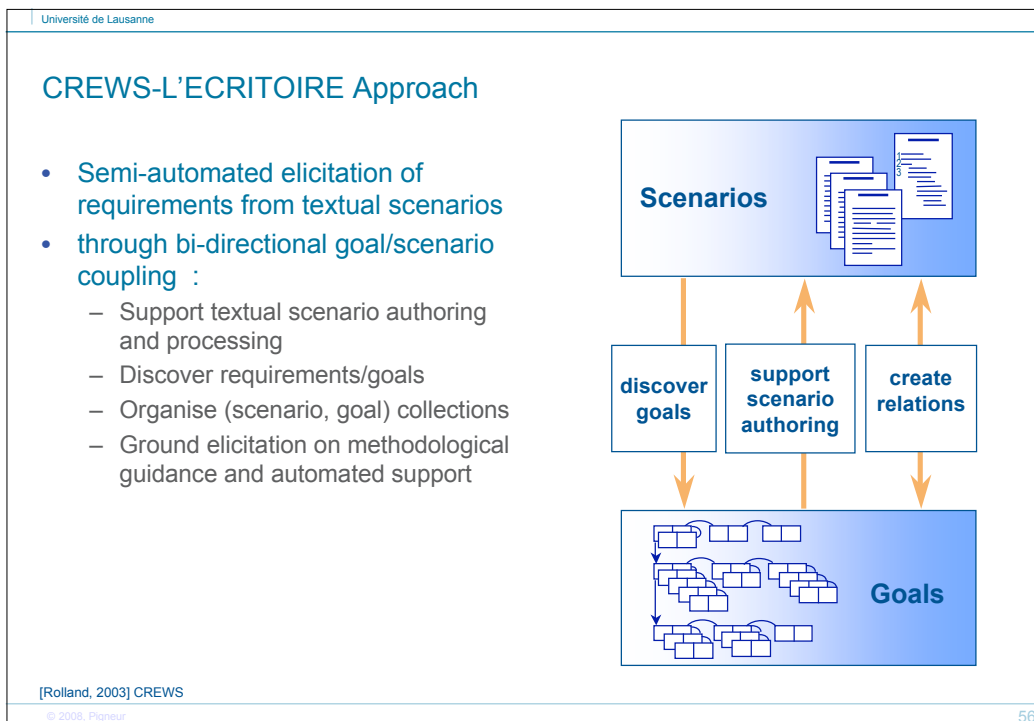
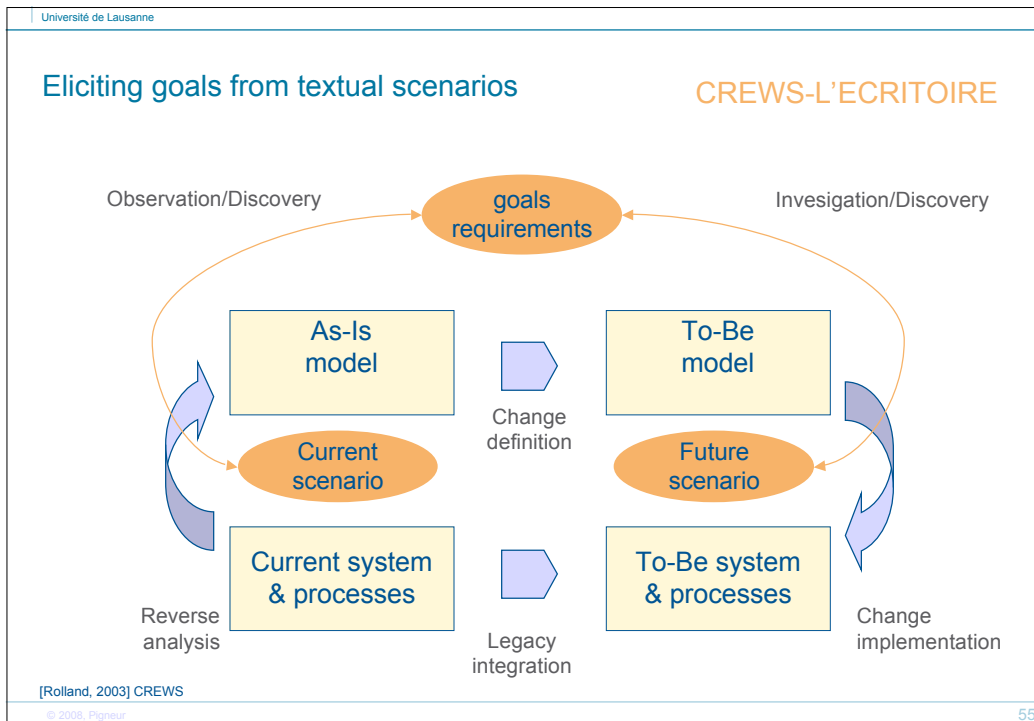
## Goal-based requirements engineering

- **Goal based reasoning is central to RE for**
  - elicitation of requirements
  - exploration of alternatives
  - exception handling
  - conflict management
- **Goals provide**
  - better abstractions for decision makers
  - support to pro-actively guide the RE process
  - pre-traceability links
- **Open issues**
  - architecture derivation
  - change management
  - monitoring of requirements
  - quality assessment

[Rolland, 2003]

© 2008, Pigneur

54



Université de Lausanne

### CREWS-L'ECRITOIRE > scenario authoring

The diagram illustrates the scenario authoring process. On the left, a person labeled 'Scenario author' is seated at a desk with a computer. An orange arrow labeled 'Authoring' points from the author to a box labeled 'Scenario'. A return arrow points from the 'Scenario' box back to the author.

*The user inserts a card in the ATM. The ATM checks the card validity and if the card is valid, the ATM displays a prompt for code to the user.....*

[Rolland, 2003] CREWS  
© 2008, Pigneur 57

Université de Lausanne

### CREWS-L'ECRITOIRE > goal elicitation

The diagram shows a 'Requirement Chunk RC1' represented as a yellow box divided into two sections: 'Scenario S1' on top and 'Goal 1' on the bottom. An orange arrow labeled 'Authoring' points from the 'Goal 1' section back to the 'Scenario S1' section.

A scenario is attached to a goal.  
A scenario is a possible behaviour described as a set of interactions between the user and the system.  
*Ex: 'Withdraw cash from ATM'*

In the forward direction  
the coupling helps making a goal concrete  
& helps detecting unrealistic, spurious goals

[Rolland, 2003] CREWS  
© 2008, Pigneur 58

Université de Lausanne

### CREWS-L'ECRITOIRE > goal discovery

The diagram illustrates the goal discovery process. On the left, a box labeled 'Requirement Chunk RC1' contains 'Scenario S1' and 'Goal 1'. An orange arrow labeled 'Authoring' points from 'Goal 1' back to 'Scenario S1'. An orange arrow labeled 'Goals discovery' points from 'Scenario S1' to a stack of three boxes, the top one being 'Goal G4.1'. Below the diagram, the text reads: 'In the backward direction the coupling helps discovering new goals'.

[Rolland, 2003] CREWS  
© 2006, Pigneur 59

Université de Lausanne

### CREWS-L'ECRITOIRE > goal organization

The diagram illustrates the goal organization process. On the left, a box labeled 'Requirement Chunk RC1' contains 'Scenario S1' and 'Goal 1'. An orange arrow labeled 'Authoring' points from 'Goal 1' back to 'Scenario S1'. An orange arrow labeled 'Discovery' points from 'Scenario S1' to a larger box on the right. This box, titled 'REQUIREMENT CHUNKs (RCs) HIERARCHY', contains three smaller boxes: 'Goal 1 | Scenario 1', 'Goal 2 | Scenario 2', and 'Goal 3 | Scenario 3'. The first two are connected by an 'AND' relationship, and the second and third are connected by an 'OR' relationship. Below the diagram, the text reads: 'REQUIREMENT CHUNKs (RCs) HIERARCHY'.

[Rolland, 2003] CREWS  
© 2006, Pigneur 60