3. Task and goal modeling

**Agenda**

- **TASK MODEL DIAGRAM**
  - Hierarchical Task Analysis (HTA)
  - GOMS
  - UAN

- **ConcurTaskTree (& CTTE tool)**

- **USE CASE**

  For your information:

  - **GOAL-BASED requirement engineering**
    - Kaos
    - j*
    - Crews-L’écritoire
Goal and task modeling

Task model

- Represents logical activities for users to reach their goal
- Can be easily discussed with both users and developers
- Two phases:
  - Task Analysis → Identify relevant tasks
  - Task Modelling → Building of the model
- Basis to generate an abstract User Interface
**Task model**

**Book Train Ticket**
- SEQ
  - Go on the web site
  - Select Train Ticket
  - Provide Customer Information
- PAR
  - Select Destination
  - Select Departing Station
  - Select hour
  - Select ticket type
- ALT
  - Select departing hour
  - Select desired arrival hour

SEQ: Sequentially
PAR: Parallelly
ALT: Alternatively

**Book Train Ticket**
- SEQ
  - Go on the web site
  - Select Train Ticket
  - Provide Customer Information
- PAR
  - Select Destination
  - Select Departing Station
  - Select hour
  - Select ticket type
- ALT
  - Select departing hour
  - Select desired arrival hour

SEQ: Sequentially
PAR: Parallelly
ALT: Alternatively

**VESALE > Task analysis**

**IHM > Analyse de la tâche**

**Structure du chapitre**

Le chapitre est structuré de la manière suivante :
- Objectif 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
  - Hétérogénéité des utilisateurs
  - Contexte de travail
  - Critères d'utilité et usabilité
  - Paramètres déscriptifs de la tâche
  - Structure de la tâche
- Techniques d'analyse
  - Choix du style d'interaction et des attributs du dialogue
  - Prédétermination du SDE en fonction du profil d'utilisateur
  - Prédétermination du SDE en fonction des paramètres descriptifs de la tâche interactive
  - Prédétermination du SDE en fonction de la complexité du traitement
  - Prédétermination des attributs du dialogue
- Exemple de paramétrage d'un dialogue - Tableau
- Sélection de SDE - exemple
  - Spécifications fonctionnelles
  - Graph de chaine et structure des fonctions
- Bibliographie

VESALE > Task analysis > 3 perspectives

- Performer perspective
- Workstation perspective
- Organizational perspective

Task

Abstract task
Planned task
Implemented task

Task analysis

Task, Users, Workstation, Activity Domain

Direct observation
Indirect observation
Oral interview
Written questionnaire
Concurrent protocol
Retrospective protocol
Use case

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]
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

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.
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

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.
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

ConcurTaskTree (CTT) temporal operators

>>> LOTOS (Language Of Temporal Ordering Specifications)

- \( T_1 \parallel T_2 \) Independent concurrency
- \( T_1 \parallel T_2 \) Choice
- \( T_1 \mathcal{W} T_2 \) Concurrency with information exchange
- \( T_1 =| T_2 \) Order independence
- \( T_1 > T_2 \) Deactivation
- \( T_1 \triangleright T_2 \) Enabling
- \( T_1 \triangleright\triangleright T_2 \) Enabling with information passing
- \( T_1 \triangleright T_2 \) Suspend-resume
- \( T_1^* \) Iteration
- \( T_1 (n) \) Finite iteration
- \( [T_1] \) Optional iteration
- \( T_1 \) Recursion

\[\text{Paternò, 2002}\]
ConcurTaskTree (CTT) - Hierarchy and temporal operators

ConcurTaskTree (CTT) - role and cooperative task model

Palanque, 2000

Paterno, 2002
**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

[Paternò, 2002]

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

**CTTE > editing a task model**

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


Use case > example (1)

Use Case 1: Buy stocks over the web

Primary Actor: Purchaser

Scope: Personal Advisors / Finance package (PAF)

Level: User goal

Shareholders 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.

Motivation:
Sufficient logging information that PAF can detect that something went wrong and can ask the user to provide details.

Success sequence:
Remote web site has acknowledged the purchase; the log and the user's portfolio are updated.

User success scenario:
1. User selects to buy stocks over the web.
2. PAF gets list of web sites to use (E: 'Frickl, Schwab, etc.) from user.
3. PAF opens web connection to the site, retaining control.
4. User browses and picks stocks 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
2b. System gets new suggestion from user, with option to cancel use case.
3a. Web failure of any sort during setup
3b. System reports failure to user with advice, backs up to previous step.
3c. User either backs out of this use case, or tries again.
4a. Computer crashes or gets switched off during purchase transaction
4b. Web site does not acknowledge purchase but puts it on delay
4c. PAF logs the delay, sets a timer to ask the user about the outcome.
4d. (see use case Update questioned purchase)
5a. Web site does not return the needed information from the purchase
5b. PAF logs the lack of information, has the user Update questioned purchase.

[Cockburn, 2000]
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]

Use case, actor, goal, and scenario

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 ...


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 fulfill 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
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 ...


---

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

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 …


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

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


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

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

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, ...)
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

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 …
### Template (I)

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

[Cockburn, 2000](http://alistair.cockburn.us/usecases/usecases.html)

### Template (II)

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

[Cockburn, 2000](http://alistair.cockburn.us/usecases/usecases.html)
Other template

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Use Case Heading &amp; Text</td>
<td>Actor</td>
<td>Use Case Level</td>
<td>Use Case Story</td>
<td>Use Case Scopes</td>
<td>Review Status</td>
</tr>
<tr>
<td>2</td>
<td>User Authentication</td>
<td>User</td>
<td>Surface</td>
<td>System Architecture</td>
<td>N</td>
<td>Self</td>
</tr>
<tr>
<td>3</td>
<td>User register</td>
<td>User</td>
<td>Surface</td>
<td>System Architecture</td>
<td>N</td>
<td>Self</td>
</tr>
<tr>
<td>4</td>
<td>User logon</td>
<td>User</td>
<td>Surface</td>
<td>System Architecture</td>
<td>N</td>
<td>Self</td>
</tr>
</tbody>
</table>

Other template

Alexander, 2004
http://www.scenarioplus.org.uk

UML diagram
two-column variation

• See Interaction design
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…”

Mission statements & strategic objectives

[van Lamsweerde, 2000] [Rolland, 2003]

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)) ≥ $1B (strategic goal)
  - Sales (VW beetle, week (24/03/2005)) ≥ 5 (operational goal)

[Mylopoulos, 2004]
Goals at different levels of abstraction

- Increase the number of contracts by 20%
- Diversify the number of ways of making offers

High level, strategic, organisation wise

Low level, tactic, design specific

Make Offer

Through the car vendor

Directly to the client by Internet

With a pull strategy

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 “satisficed”
    - Soft goal fulfilment os relative and “good enough”, rather than absolute and minimal
  - User-friendly [Interface]
Eliciting goals

- Bottom-up: WHY?
- Top-down: HOW?

Modelling goals (AND)

- AND link: the father goal is satisfied only if the children goals are satisfied
  - AND links help in driving the process of goal operationalisation
Modelling goals (OR)

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

![Diagram showing OR refinement]

Goals links and dependencies

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

![Diagram showing goals links and dependencies]
Types of goals

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

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
Formalization of goals

Achieve [BookRequestSatisfied]

(\forall \text{bor: Borrower, } b: \text{Book, lib: Library})
Requesting (\text{bor, } b) \land b.\text{subject} \in \text{lib.coverageArea}

\Rightarrow \Diamond (\exists \text{bc:BookCopy}) ((\text{Copy(bc, } b) \land \text{Borrowing(bor, } bc))

Q: property will hold in all future states
\Box Q: property will hold in some future states

Goals, stakeholders and conflicts

- Goals may be owned by stakeholders
  - "train frequency increased" (passengers)
  - "number of passengers increased" (train company)

- Achieving goals require cooperation among agents/actors
  - DEPENDENCIES < actor1, goal g, actor2>
  - goal g requires the cooperation between actor1 (dependee) and actor2 (dependent)

- Potential for conflicting viewpoints
  - "card kept when KO-customer" (banks)
  - "card returned when KO-customer" (customers)
Social models: actors & dependencies

- Social relationships among actors

1. Work action approach
   - speech act, …

2. Social dependency approach (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

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] [Mylopoulos, 2004] [Yu, 1995]
Eliciting goals from textual scenarios

- **As-Is model**
- **Current scenario**
- **Current system & processes**

- **To-Be model**
- **Future scenario**
- **To-Be system & processes**

**goals requirements**

Observation/Discovery

Investigation/Discovery

Reverse analysis

Legacy integration

Change definition

Change implementation

[Rolland, 2003] CREWS-L’ECRITOIRE

**CREWS-L’ECRITOIRE Approach**

- Semi-automated elicitation of requirements from textual scenarios

  - through bi-directional goal/scenario coupling:
    - Support textual scenario authoring and processing
    - Discover requirements/goals
    - Organise (scenario, goal) collections
    - Ground elicitation on methodological guidance and automated support
CREWS-L’ECRITOIRE > scenario authoring

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.

CREWS-L’ECRITOIRE > goal elicitation

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
CREWS-L'ECRITOIRE > goal discovery

Requirement Chunk
RC1

Authoring
Scenario S1
Goal 1

Goals discovery
Goal G4.1

In the backward direction
the coupling helps discovering new goals

[Rolland, 2003] CREWS

CREWS-L'ECRITOIRE > goal organization

Goal organization
Scenario S1
Goal 1

Discovery

Authoring

AND
Goal 1
Scenario 1
Goal 2
Scenario 2
Goal 3
Scenario 3

OR

REQUIREMENT CHUNKs (RCs)
HIERARCHY

[Rolland, 2003] CREWS