Linking User Task and User Interfaces: a Model-based Approach

Fabio Paternò
fabio.paterno@isti.cnr.it
http://giove.isti.cnr.it/~fabio/

ISTI-C.N.R.
Pisa, Italy
Goals of this part of the summer school

- Design an interactive systems paying attention to users and tasks they aim to accomplish;
- Set up a task analysis for a specific project, and derive a task model;
- Use task models for design and development;
- Design interfaces for applications that can be accessed through various types of devices and modalities.
This morning lesson

- Introduction to usability
- Task analysis and modelling
- Tools for task modelling
- Task-based design
What is usability

- Multi-dimension concept
- Satisfaction, efficiency, easy of learning, safety, flexibility, ...
- The importance of each dimension depends on the application domain
- Example: air traffic control, game, bank office
Usability benefits

- Increase efficiency
- Increase productivity
- Reduce errors/increase safety
- Reduce training
- Reduce support
- Increase acceptance
- Decrease development cost
- Increase sells
When to consider usability

- Need for an Iterative Method
- Initial User Involvement
- Design (scenarios, task analysis, …)
- Prototype
- Test of the complete system
Norman’s cycle of interaction

Goal

Intention

Action

Perception

Execution

Evaluation

Interpretation

Evaluation

User

Computer
Detectable problems

- Lack of correspondence between user intentions and actions supported by the interface
- Lack of correspondence between representations provided by the system and those expected by the user
- The best interface is invisible; it does not pose obstacles when users perform their tasks
Some Basic Principles

- Focus on users and their tasks
  - Not on the technology
- Consider first functionality, then how to present them
  - Need for a conceptual model
- Conform to the user’s view of tasks
  - Natural interaction, user language, avoid internal detail
- Do not complicate user tasks
  - Ease frequent tasks, not add complications
- Facilitate learning to use
  - Avoid ambiguity, consistency
- Provide information, not data
- Test with real users
Some Basic Metrics

- Time to complete a specific task
- Number of tasks the user has completed in a given amount of time
- Number of tasks that the user has not been able to complete
- Number of errors
- Time spent in recovering from errors
- Number of errors wrt correct interactions
- Number of tasks performed and never performed
- Number of features recalled by the users after the session
- Frequency of use of manuals or help system
Approaches to usability evaluation

- Evaluation based on user tests in laboratory
- Evaluation based on field observation
- Evaluation based on user feedback (e.g. interviews, questionnaires)
- Remote evaluation
- Model-based evaluation (e.g. GOMS, simulation)
- Inspection-based evaluation (e.g. heuristic evaluation, cognitive walkthrough)
Motivations for task analysis and modelling

- Main usability principle:
  - Focus on the users and their tasks

- Tasks represent the logical activities performed for reaching user goals

- Need for modelling is most acutely felt when the design aims to support system implementation as well
  - Especially for large projects and some application domains
Main Phases in the Design & Development Cycle

- Requirements
- Conceptual design
- Detailed design
- Implementation
- Evaluation
Significant Models in HCI

Models
- Task & Object Models
- Interactor
- User
- Platform
- Environment

Design time
- Task and object model
- Abstract User Interface
- Concrete User Interface

Multiple-Version

Run time
- Adaptivity
  - Run-time events
  - Reconfiguration

User Interface
Tools in Model-based environments

- Modelling Tools
- Analysis Tools
- Development Tools
- Reverse Engineering Tools

Models
- Task & Object Models
- Abstract UI
- Concrete UI
- Context

Designer
- Design criteria
- Transformations

User Interface
- Context of use

Development Tools

Reverse Engineering Tools
Definitions

- Task – activity that has to be performed to reach a goal
- Goal
  - desired modification of state
  - Attempt to receive state information
- Each task is associated with one goal
- Each goal is associated with one or multiple tasks
- Multiple abstraction levels - Basic task
- Task Analysis
- Task Models
Scenarios

- Informal, compact description of:
- one (or multiple) specific user
- Who interacts with a specific interface
- To reach a specific goal
- In a specific environment
Silvia is looking for interesting papers on patterns. She makes a request to the on-line library by giving the name of the topic as parameters of her request, and indicating that she is interested in papers written in English. The order of providing these two parameters is not important. She receives a long list of references. As she is interested in recent contributions, she adds a further constraint in the request so that she receives information only on papers published in the last five years. The new list of publications is more manageable. She understands that the works by Gamma are very relevant. She would like to have them grouped so that they are presented together. Thus she makes a new request adding the constraint that the author has to be Gamma. The result is the information that she was looking for. Now she can move to another request for another topic.
Use of Scenarios

- Capture the context where the application is used
- Elicite requirements
- Identify important episodes from the user behaviour
- Provide a context for performing evaluation
- Highlight issues and stimulate discussion while requiring limited effort to develop
Claim analysis (Carroll)

- Some design feature
- + causes (desirable consequences)
- - causes (undesirable consequences)

- Video information
- + is a very rich, intrinsically appealing medium
- - is difficult to search, and must be viewed linearly in real time
Scenarios in the Design Cycle

- Requirements
- Conceptual design
- Detailed design
- Implementation
- Evaluation
- Scenarios
Use Cases

- Purpose
- Content
- Plurality
- Structure
Task Analysis

- Identification of the tasks that should be supported
- They can be structured hierarchically
- Identification of the objects that are necessary to perform such activities
- Task allocation between the user and the system
- Knowledge required to accomplish the task
Task analysis

Esemple: Task analysis of tourists visiting a virtual museum application

- Tourists are characterised by a low average knowledge of the topics considered. Usually they prefer to have guided tours through the rooms of the museum and the town with pictures and information about the works of art. However linear pre-defined tours alone would be too restrictive so some degree of navigational freedom is important. Access to the information is provided with the support of spatial representations: the museum and town maps. This allows users to have immediate information about the locations of the works.

- Tourists want general information on the artistic works, and this information has to be presented clearly and in a limited amount because it has to be interpreted easily. Thus a work will be presented by an image, the title, a short description, the name of the author, the material and technique used for its creation, and when it was made. Additional information about the museum and the town can be provided on request, such as the path to get to the museum from the closest railway station or airport, information (title, data, location) on further exhibitions, and historical information on the town and the museum.
Task Analysis (task list)

- Access to guided visits to the museum and the town
- The system supports some degree of freedom in the navigation
- Access to the information through spatial representations
- Access to general information regarding works of art
- The system presents limited information in a clear manner
- The system presents a work of art through an image, the title, short description, author, material and technique applied, date
- Additional information regarding the museum and the town can be provided on request
Techniques to support task analysis

- Interviews, workshops, …
- Questionnaires
- User observations
- Analysis of how activities are performed
- Analysis of existing documentations and how users are trained.
Moving from informal to structured representations

Informal Analysis
- Interviews
- Questionnaires
- Existing documentation
- Activity analysis
- Current training

Structured Analysis
- Requirements identification
- Scenarios
- Use cases
- Task analysis

Abstractions
- Domain models
- Task models
- User and System Models
- Dialogue models
- Properties
Task Analysis in the Design Cycle

Task Analysis

Requirements

Conceptual design

Detailed design

Implementation

Evaluation
U-Tel for a Logistics Domain Example

Informal text provided by end user, domain expert, or our notes

Lists of objects, tasks, and user types

Informal task model
Natural development

- From easy-to-use applications to applications easy-to-develop and personalise
  - Familiar and intuitive representations vs precise descriptions

- Extension of the model-based approach
  - Integration of informal and structured representations
  - Provide effective representations
  - Support for different entry-points and abstraction levels
Integration of informal and structured specifications

- Sketches
  - Improve the communication among people
  - Focus more on the idea rather than on the detail
Integration of informal and structured specifications

- MultiModal Interaction
  - Vocal
    - Use of keywords and sentences
  - Input device (keyboard, mouse, stylus,..)

At the beginning the user inserts name and password in order to access the system; then the system visualises the personal page with information on areas of interest.
CTTE support for task identification

Linking User Task and User Interfaces: A Model-Based Approach
Tool support to structure the task model

Scenario

Roles Specification | Cooperative Tasks Specification | Task Model Builder

Tasks List

- **Ground**
  - guide the departing aircraft
  - communicate to the pilot the tower radio
  - detecting wrong positions
  - gives the pilot of the AZA 1645 the instructions... 

Task Description

... from the apron area up to the holding position on the departure runway (give the path)
Use of Task Models

- Improve understanding of the application domain
- Record the result of interdisciplinary discussion
- Support effective design
- Support usability evaluation
- Support the user during a session
- Documentation
Task modelling

- Flexible and expressive notations with precise semantics
- Systematic methods able to indicate how to use information in the task models
- Availability of automatic tools to manipulate and use such information efficiently
Advantages of Task-based approaches

- For the designer: high-level, structured approaches which allow an integrated approach to both functional and interactional aspects
- For the end user: support the generation of more understandable systems
The many possible task models

- (Describe) Existing System, to better understand the underlying design and analyse its potential limitations and how to overcome them
- (Define) Envisioned System, to indicate how activities should be performed in order to obtain a new, usable system
- User, how users think that the activities should be performed
What should task models represent

- Task models can be represented at various abstraction levels.
- The main high-level tasks are considered in requirements analysis.
- When designers aim to provide precise design indications then the activities are represented at a small granularity.
- The subject of a task model can be either an entire application or one of its parts.
- The application can be either a complete, running interactive system or a prototype under development.
Task Models vs Scenarios

- Scenarios are informal descriptions of a specific use in a specific context.
- Task models describe the main possible activities and their relations.
- Scenarios can support task model development.
- Task models can support scenarios identification.
Task Models in the Design Cycle

Requirements

Evaluation

Conceptual design

Task Models

Implementation

Detailed design
Representations of Task Models

- Different syntax (textual vs graphical)
- Different level of formality
- Different set of operators for task composition
Hierarchical Task Analysis
GOMS: Goals, Operators, Methods and Selection Rules

<table>
<thead>
<tr>
<th>GOAL: EDIT-MANUSCRIPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>GOAL: EDIT-UNIT-TASK repeat until no more unit tasks</td>
</tr>
<tr>
<td>GOAL: ACQUIRE-UNIT-TASK</td>
</tr>
<tr>
<td>GET-NEXT-PAGE if at end of manuscript</td>
</tr>
<tr>
<td>GET-NEXT-TASK</td>
</tr>
<tr>
<td>GOAL: EXECUTE-UNIT-TASK</td>
</tr>
<tr>
<td>GOAL: LOCATE-LINE</td>
</tr>
<tr>
<td>[select: USE-QS-METHOD</td>
</tr>
<tr>
<td>USE-LF-METHOD]</td>
</tr>
<tr>
<td>GOAL: MODIFY-TEXT</td>
</tr>
<tr>
<td>[select: USE-S-METHOD</td>
</tr>
<tr>
<td>USE-M-METHOD]</td>
</tr>
<tr>
<td>VERIFY-EDIT</td>
</tr>
</tbody>
</table>
Limitations of GOMS

- It does not consider user errors
- It does not consider the possibility of interruptions
- It considers only sequential tasks
- It can be inadequate for distributed applications (such as web-based applications)
Task Model Representations

GTA
- Main Goal
  - Task 1.1
  - Task 1.2
  - Task 2.1
  - Task 2.2
  - SEQ

CTT
- Task 0
  - Task 0.1
  - Task 0.2
  - Task 0.2.1
  - Task 0.2.2

MAD
- Task 0
  - SEQ
  - Task 0.1
  - Task 0.2
  - Task 0.2.1
  - Task 0.2.2

UAN
- Task 0 = Task 0.1
- \{Task 0.2.1 | Task 0.2.2\}

OpTA
- SEQ
  - Task 0.1
  - ALT
    - Task 0.2.1
    - Task 0.2.2
**Domain object model**

- A domain object model defines the objects that a user can view, access, and manipulate through a user interface.
- A domain object model represents objects of the domain with their relationships.
- Historically, data models have been considered for a while, but they are only a subset of domain models.
Use of task models in the design cycle

- Requirements of The New System
- Task Modelling
- Existing System's Task Model
- Envisioned System's Task Model
- Design
- Early System's Prototype
- Current Prototype's Task Model
- Evaluation
- Reverse Modelling
- Design And Software Development
- Engineered Prototype

Linking User Task and User Interfaces: A Model-Based Approach
Linking User Task and User Interfaces: A Model-Based Approach

Possible Integration
UML/Task Models

Use Cases

Domain Model

Task Model

Scenarios

Integration

Abstract User Interface Model

User Interface 1

......

User Interface N
ConcurTaskTrees

- Focus on Activities
- Hierarchical Structure
- Graphical Syntax
- Rich set of temporal operators
- Task allocation
- Objects and task attributes
Categories of tasks

- interaction
- application
- user
- abstract
### Temporal operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Enabling</strong></td>
<td>$T_1 \gg T_2$ or $T_1 [ ] \gg T_2$</td>
</tr>
<tr>
<td><strong>Disabling</strong></td>
<td>$T_1 [&gt; T_2$</td>
</tr>
<tr>
<td><strong>Interruption</strong></td>
<td>$T_1</td>
</tr>
<tr>
<td><strong>Choice</strong></td>
<td>$T_1 [ ] T_2$</td>
</tr>
<tr>
<td><strong>Iteration</strong></td>
<td>$T_1^*$ or $T_1^{(n)}$</td>
</tr>
<tr>
<td><strong>Concurrency</strong></td>
<td>$T_1 || T_2$ or $T_1</td>
</tr>
<tr>
<td><strong>Optionality</strong></td>
<td>$[T]$</td>
</tr>
<tr>
<td><strong>Order Independency</strong></td>
<td>$T_1</td>
</tr>
</tbody>
</table>
The ConcurTaskTrees Notation

Hierarchical structure

Temporal relations

Task Allocation
Task and attributes

**Interaction tasks**
- Selection
- Edit
- Control
...

**Application task**
- Overview
- Feedback
- Generating alerts
- Grouping
...

Generating alerts
Criticisms of Task Model-based Approaches

- The benefits of adopting task modelling do not justify the extra-time requested
- Task models are not applicable to creative activities
- If the task hierarchy is directly implemented then we obtain modal interfaces
- Task-based approaches lead to verb-noun style interfaces
Tool support

• How task models are represented
• How to construct the task model
• How task models can be manipulated
• What metrics can be useful to analyse the models
• How interactive simulation can be useful for designers
• What other type of interactive analysis can be useful
Tool Support in CTTE

- Flexible editing of the task model
- Using informal descriptions in modelling
- Checking completeness of the specification
- Saving the specification in various formats
- Simulating the task model
- Comparing task models
- Running scenarios
- http://giove.isti.cnr.it/ctte.html
The user interface of the CTTE tool
Automatic expansion of task patterns
Example of automatic task path identification
Checking completeness of the specification
Comparison of Task Models
Interactive Simulator

- Designers can check the specified behaviour
- It can support a multidisciplinary discussion on design decisions at the task level;
- It can be employed as interactive documentation of an application to explain to end-users how to use it
Simulating dynamic behaviour
From Task Model to Abstract Scenario
The user interface of the CTTE tool - DEMO
How to ease task modelling?

- Reverse engineering (for example, WebRevEnge)
- Using interaction logs
- Using informal representations (scenarios, sketches, …)
- Why not using information visualization techniques?
New Interface: Model Editor
The new representation

- Semantic zoom
- Word wrap
- Compact arcs
- New Icons
- Detail
- Grouping of sub-tasks
- Brackets to indicate priorities
- Second level
- Fisheye
- Focus center
- New overview
The fisheye approach applied to simulator output
Modelling Multi-User Applications

User1

User2

User3

Cooperative part
Cooperative aspects
Show example task model multi-user application
Limitations of current approaches in UI design

- Design completely based on ad hoc solutions and the intuition of the designer
- Visual tools do not support mapping between logical activities and UI elements
- UML and its tools are oriented to the system design
- Lack of support for user task-based design
Concrete design process

- Mapping abstract elements (tasks, domain) to concrete elements (interaction techniques, windows)
- Partitioning tasks among windows and/or media
- Defining composition of elements in a presentation (laying out elements, ...)
- Applying ergonomic rules
Organizing the user interface structure

- Group elements
- Create hierarchies
- Represent relations
- Overall balance
Possible Views of an Interactive System

- Task and object – *I want to select a work of art*
- Abstract Interface – *Single selection object with high cardinality*
- Concrete Interface – *List Interaction object with X elements*
- Code – *List object in Java or XHTML or* ....
Task model - based design

1. Analysis of temporal operators amongst tasks
   - Identify structure of dialogues

2. Analysis of each task (objects, attributes, ...)
   - Choose suitable interaction objects
Deriving information from operators
Enabled Task Set

• Identify sets of task that are enabled over the same period of time

• Their identification depends on the logical and temporal structure of tasks in the model

• Useful for identifying the presentations that can be chosen to support sets of tasks

• **How many presentations?** One, the total number of enabled task sets, an intermediate value
Example
Presentation Task Sets

Presentation Task Sets Generation

List of Presentation Task Sets
- Source: Presentation Task Set 1: (Show_intro, Select_access_to_gen_info, Select_access,...)
- Destination: Transformations applied on Source frame, are shown in this Destination frame.

Save PTSs (source) in XML
Show previous or current PTSs in source: No Criteria
Apply >>
Move <<
Close

Select one Criterion:
- 'Joining when Enabling'
- 'Single Element Sets'
- 'Sharing most elements Sets'
- 'Exchanging information'

If two PTSs differ for only one element, and those elements are at the same level connected with an enabling operator, they are joined together.

If a PTS is composed of just one element, it is joined with another PTS.

If some PTSs share most element they are unified.

If there is an exchange of information between two tasks, they are included in the same presentation in order to highlight such information transfer.
Useful information in the task model

- Identify groups of tasks that share the same parent tasks.
- Low-level tasks that share the same parent are located close to each other.
- Concurrent communicating tasks (||[]) are supported by close elements.
- Interface elements corresponding to disabling tasks (>[>) are located in such a way to highlight their role.
From the Model to the Interface
Task-interface relations
Tasks-interface relations

Linking User Task and User Interfaces: A Model-Based Approach
Tasks-interface relations
Grouping tasks that share the same parent task

Communicating concurrent tasks (||[|]|) are presented close to each other