User Interface Design: From Work Tasks to Interactive System Designs

(Tutorial Notes)

Technical Report No. 693, November 1994

Stephanie Wilson and Peter Johnson

Human-Computer Interaction Laboratory
Department of Computer Science
Queen Mary & Westfield College
University of London

steph, pete@dcs.qmw.ac.uk

Abstract

Recent work in the field of model-based user interface design offers techniques for creating and exploring user interface designs at a higher level of abstraction than is possible using traditional user interface toolkits and interface builders. The user interface designer develops a model of the proposed interactive system and expresses the model using a high-level notation; an executable prototype is then generated automatically from the model. One subclass of the model-based approaches may be characterised as task-based user interface design techniques. These techniques offer methods for creating design solutions based on information concerning the users' tasks: they provide a systematic approach that extends user interface design beyond "hacking".

This technical report contains the notes for a tutorial on task-based user interface design which was originally presented at the British HCI'94 conference. The tutorial reviews the field of model-based interface design and assesses the advantages and disadvantages of the approach. It also introduces and examines key concepts of task-based design and discusses the role of task analysis techniques. One recent approach to task-based design, Adept, is studied in greater depth. (Participants at the tutorial had the opportunity to gain practical experience in task-based design and in the use of the Adept tools through a design case study.) The tutorial concludes with an assessment of future directions for task-based design.

Research area: Human-computer interaction

Keywords: User interface design environments, task-based design, task analysis, design methods

Table of Contents

User Interface Design: From Work Tasks to Interactive Systems

HCI'94 Tutorial

Stephanie Wilson and Peter Johnson

Queen Mary and Westfield College

University of London

steph, pete@dcs.qmw.ac.uk

Agenda

- Introduction — objectives
- The model-based user interface design paradigm
- Task-based design in Adept
- Task analysis
- Introduction to case study
- Case study:
  - Task model
  - Task design
  - Abstract interaction model
  - Interface generation
  - Design presentations
- Conclusions

1. Objectives

- To introduce and examine task-based and model-based approaches to user interface design
- To review a selection of software environments that support these design paradigms
- To provide practical experience in the task-based design of an interactive system

2. The Model-Based Design Paradigm

- Introduction — objectives
- The model-based user interface design paradigm
- Task-based design in Adept
- Task analysis
- Introduction to case study
- Case study:
  - Task model
  - Task design
  - Abstract interaction model
  - Interface generation
  - Design presentations
- Conclusions

What is Model-Based Design?

- A technique whereby interface designs are created and expressed at a high-level of abstraction
- Designs expressed as 'models' using notations whose constructs are meaningful in terms of HCI
- Models variously referred to as 'Application Models' or 'Interaction Models'
- Automatic generator tools transform high-level designs into executable user interfaces, mediated by design guidelines
- Comparisons with User Interface Management Systems

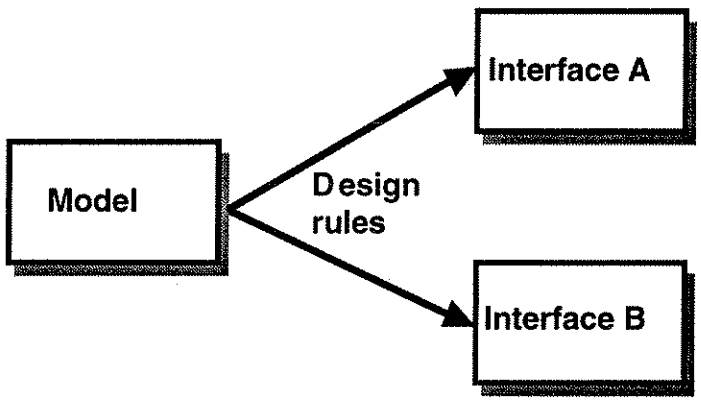
Motivations (1)

- Provide high-level representation of designs
- Provide easier construction of designs:
  - reduce overheads of low-level coding using user interface toolkits or interface builder tools
  - support interface generation
  - provide easier modifiability of designs
  - support reuse of designs
- Provide methodological support:
  - improve support for design life-cycle
  - facilitate early exploration of design alternatives

Motivations (2)

- Support reasoning about designs
- Extend run-time functionality by migrating models to the run-time environment, for example:
  - improve help
  - offer explanation facilities
  - provide user tailoring and configuration
- Provide design traceability
- Provide smarter design tools
- Improve "usability"

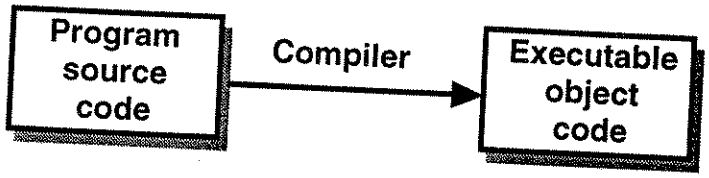
The Model-Based Paradigm



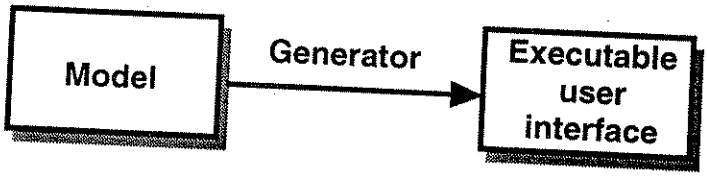
Different interfaces generated from the same model, according to design rules

Programming Language Analogy (1)

Program design:



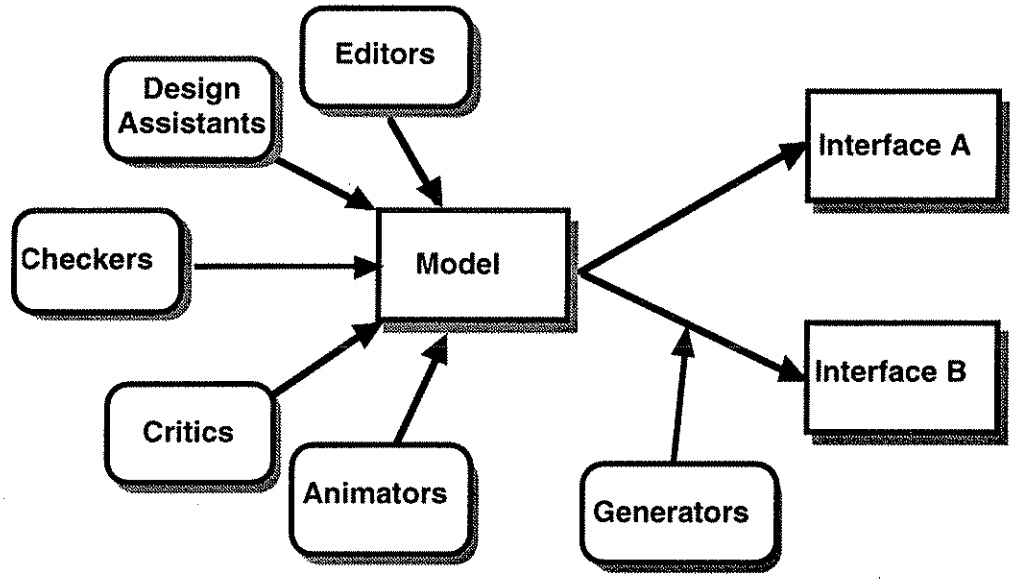
Model-based interface design:



Programming Language Analogy (2)

- Programs and Models express high-level design solutions
- Compilers and generators automatically produce low-level executable representations
- Interface modelling notation can be regarded as a high-level user interface programming language

Tools in the Design Environment



Modified from [Puerta and Szekely 94, p.7]

What is Modelled?

Some or all of:

- **Domain objects and actions**
- **Dialogue and interface behaviour**
- **Presentation**
- **Implementation platform**
- **User characteristics**
- **Tasks**

UIDE

User Interface Development Environment [Foley et al 91]

- Application model — defines the allowed user actions and their effects on application data objects
- Interface model — defines interface behaviour that accomplishes the user actions in terms of interface objects
- Pre- and post-conditions associated with the actions
- Objects defined in terms of their attributes

ITS

[Wiecha et al 90]

- Actions — (application model) procedures that perform computations, data model
- Dialogue — specification of control flow between frames
- Style rules — construction of displays, screen layout, consistent usage of widgets
- Display objects - graphical toolkit building blocks

HUMANOID

[Szekely et al 92, 93]

- Application semantics — inputs, commands (actions) and groupings of these
- Presentation — hierarchical decomposition of display, pluggable components, interaction style rules
- Behaviour — interactors with their state, effects and activity
- Dialogue — sequencing described implicitly in application model, e.g. through pre-conditions on commands

ADEPT

Advanced Design Environment for Prototyping with Task models

[Johnson et al 93], [Wilson et al 93]

- Extant Task, Designed Task — goal structure, task procedures and actions, task objects
- Abstract Interaction Model — Classes and groupings of interaction objects, dialogue structure
- Concrete Interaction Model — Interaction objects, dialogue, screen layout
- Design guidelines — allocation of interaction styles, widget selection, screen layout

Properties of Environments (1)

- Focus of Environment:
  - rich application model — UIDE, Humanoid, ITS
  - process of design — Adept
- Available at runtime:
  - UIDE, Humanoid, ITS
- Supports top-down and bottom-up design:
  - UIDE

Properties of Environments (2)

- Usable by non-programmers:
  - Adept, UIDE, Humanoid, ITS
- Generality:
  - all are limited to WIMP style interfaces or worse
  - all are restricted to one or few graphical toolkits
- Availability
  - none are widely available, but ITS is a production quality tool

Advantages

- Rapid prototyping through early exploration of design alternatives
- Improved reusability and modifiability of designs
- Design 'audit trails'
- Advanced design tools
- Consistency of interfaces
- Methodology: support for conceptual level design (or getting there)

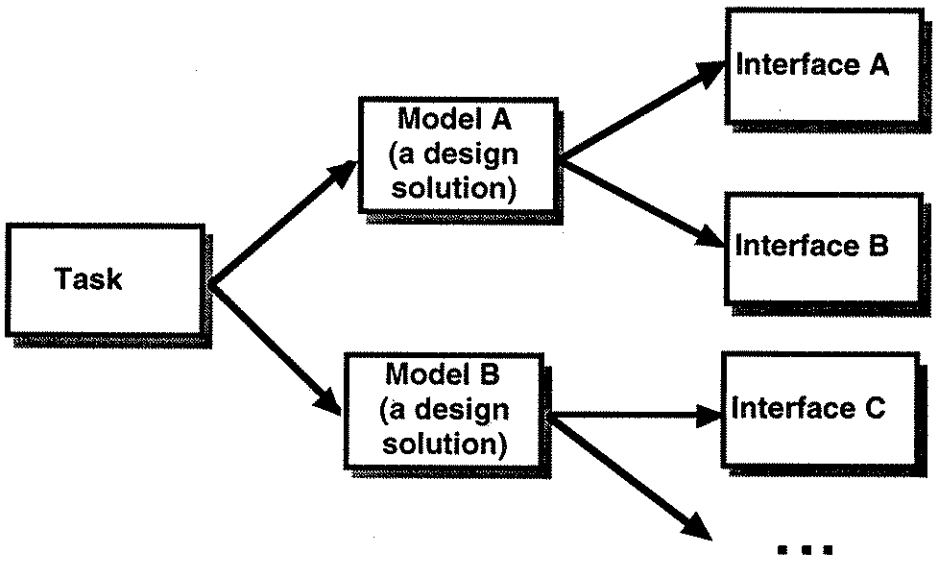
Disadvantages

- Effort of producing models
- Tendency to consider only top-down design processes
- Less flexible than coding, preventing designers from producing novel solutions
- Usability issues — most model-based environments are research tools and still in their infancy
- Focus on representing and implementing design solutions, ignoring other aspects of the design process
- Design solutions can only be as good as the environments in which they are created — unlikely to be state-of-the-art technology

Task-Based Design Aims

- Extend basic model-based design paradigm to include other aspects of the design life-cycle
- Integrate human factors and software engineering approaches to interactive system design
- Support user-centred design by basing design solutions on descriptions of the users' tasks
- Provide an integrated design support environment

Task-Based Design Paradigm



Alternative design solutions may support the same task

MUSE

Method for USability Engineering [Lim et al 92, 94]

- Integration of human factors techniques with the Jackson System Development method
- Task descriptions of extant and target systems
- Task hierarchies described using the Jackson structured system design notation
- Points of communication between system design and interaction design identified
- Complex method, but no tool support as yet

TRIDENT

[Vanderdonckt 93], [Bodart et al 94]

- Integrated, task-based methodology with tool support for part of the process
- Models tasks, applications, dialogue, presentation and interaction styles
- Plan-oriented approach to task modelling, similar to TKS
- Automatic generation of user interfaces from abstract models, directed by design rules

DIGIS

Direct Interactive Generation of Interactive Systems (de Bruin et al '94)

- Task-based methodology for developing user interfaces, with partial tool support
- Integrates task analysis and object-oriented analysis through a 'Domain Application Model'
- Models the application (as objects, actions, relationships), the system task model and user task model
- Partially automated interface generation, with direct manipulation editing to describe presentation and dialogue
- Run-time architecture based on PAC

3. Task-Based Design in Adept

- Introduction
- The model-based user interface design paradigm
- Task-based design in Adept
- Task analysis
- Introduction to case study
- Case study:
  - Task model
  - Task design
  - Abstract interaction model
  - Interface generation
  - Design presentations
- Conclusions

Adept Motivations

(Advanced Design Environment for Prototyping with Task Models)

- User-centred design should include considerations of the users and their tasks
- Yet, the human factors contribution (e.g. task and user models) is not always made explicit and therefore relies on designer's intuition to influence the design
- Failure of current tools (e.g. toolkits, interface builders) to support anything beyond construction and prototyping of the user interface software

Adept Aims

- Investigate how models of users' tasks can contribute to the creation and development of a system design
- Provide an integrated, model-based environment for the support of user interface design and development
- Increase involvement, communication and participation by designers, users and organisations

Philosophy underlying Adept: design practice will be improved and 'better' user interfaces will result from basing designs on information about the users' tasks

Features

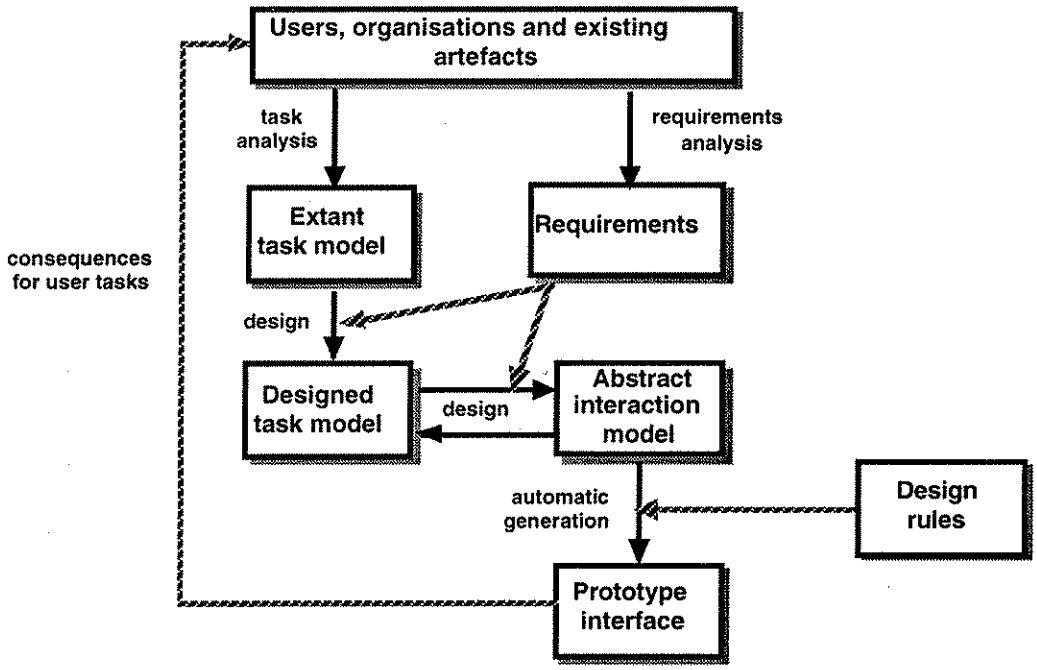
- Iterative, task-based design process
- Integrated design environment supporting all stages of design including task modelling
- Generation of user interfaces from interaction models
- Design activities may proceed at any level of abstraction (i.e. not necessarily top-down design)
- Simplicity in notations and tools to facilitate ease of learning and ease of use
- All activities centred around a library of models

Aspects

Three important aspects:

- Design life-cycle — encompassing analysis procedures and design processes
- Models — expressing the results of analyses and design
- Tools — supporting the models and design life-cycle

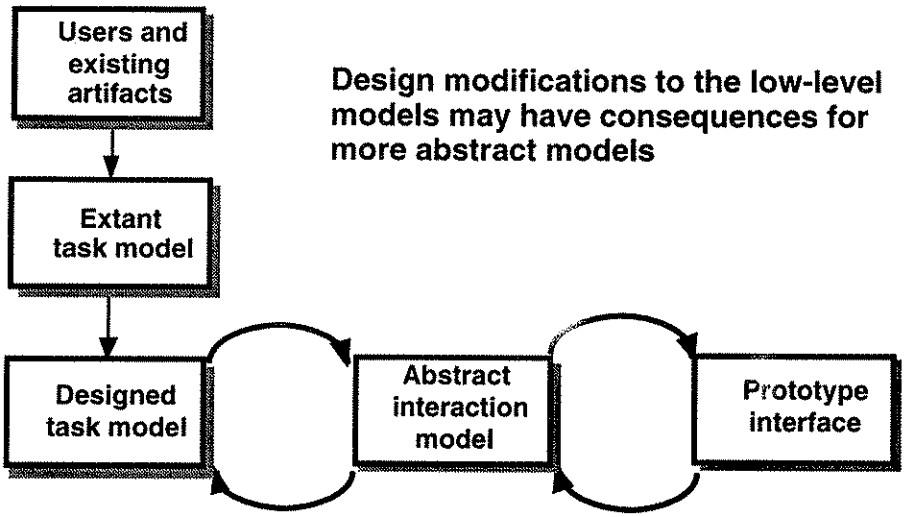
Design Life-Cycle Overview



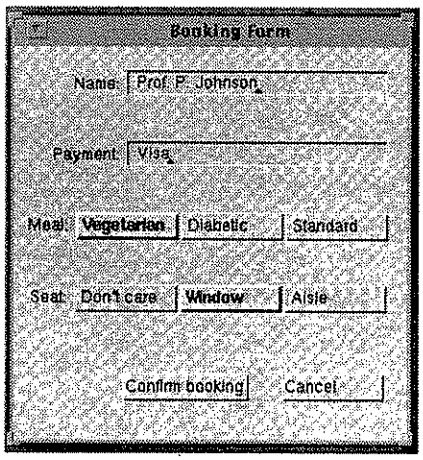
Adept Components

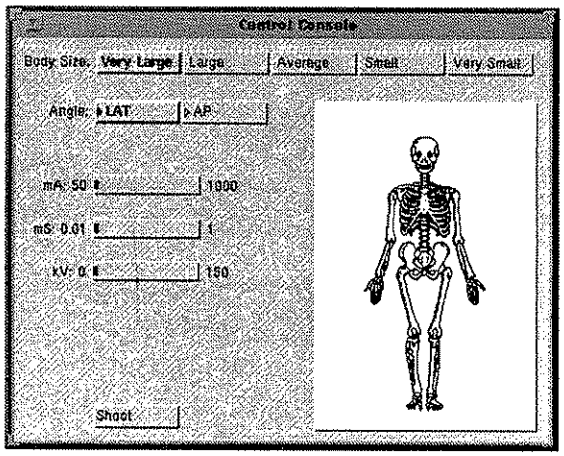
- Extant Task Model — represents the knowledge possessed by users and recruited in performing an existing task
- Designed Task Model — represents some proposed new task situation, without prior commitment to detailed design or technology
- Abstract Interaction Model — provides a high-level description of the user interface (conceptual level design)
- Design Rules — encapsulate user interface design expertise as a set of design guidelines
- Executable Interface — provides a prototype interface composed of widgets from a pre-defined widget set

Simplified Adept Design Process



Interface Examples





Demonstrator

- Illustrates the Adept approach to model-based design
- Includes prototype implementations of the Adept models and tools
- But lags behind our understanding and use of the models, e.g. is limited to supporting the creation of interfaces comprised from a set of standard components

Major Case Study

- Aim: to demonstrate the applicability of the Adept models and tools
- Realised through design of a system to support the task of taking X-rays for use in clinical diagnosis
- Task analysis carried out with radiographers from the London Hospital and task model constructed
- Task model provided the starting point for the design of a new X-ray control console

(Further details of case study are provided in the appendix to the tutorial notes)

4. Task Analysis

- Introduction
- The model-based user interface design paradigm
- Task-based design in Adept
- Task analysis
- Introduction to case study
- Case study:
  - Task model
  - Task design
  - Abstract interaction model
  - Interface generation
  - Design presentations
- Conclusions

What is Task Analysis?

- Analysis of work and jobs
- Applied to design and evaluation of training, jobs and work, equipment
- Purposes:
  - Describe
  - Explain
  - Diagnose
  - Prescribe
  - Predict
- Explicit and Systematic

  or
- Implicit and Intuitive

What to Analyse?

- Behaviours
- Cognitive processes
- Performance
- Knowledge
- Organisation
- Social contexts

Distinctions

- Distinguish between:
  - Methods of collecting information (data)
  - Methods of analysis
  - Methods of use

- Distinguish between:
  - Task Analysis using TKS
  - Adept modelling and design
  - Formal description of tasks or design

Examples Include:

- Hierarchical Task Analysis [Annett et al 71]
- Task Analysis for Knowledge Description [Diaper 89]
- Task Knowledge Structures [Johnson 92]
- Goals, Operators, Methods, Selectors [Card et al 83]
- Task-Action Grammar [Payne & Green 86]
- Extended Task-Action Grammar [Tauber 90]
- User Action Notation [Siochi & Hartson 89]

Hierarchical Task Analysis (HTA)

[Annett et al 71]

- Decomposes tasks into sequences of subtasks at multiple levels of detail
- Used widely in training and process control equipment design and evaluation

Task Analysis for Knowledge Descriptions

[Johnson et al 94], [Diaper 89]

- Developed to identify training needs and subsequently applied to HCI design
- Identifies action and objects associated with tasks
- Distinguishes between specific and generic
- Identifies generic action/object "grammar"

Understanding Work and HCI

- GOMS, TAG and the like try to understand human-computer interaction tasks at a "micro-level" and not at a work level
- HTA, TAKD and TKS try to understand work and apply this to human-computer interaction

Describing Work Tasks

- Existing work task situations
- Proposed work task situations
- Providing basis for design and evaluation in terms of effects on work tasks

Identifying Work Tasks

- Existing tasks are defined by workers and workplace
- Proposed tasks are hypothesised effects of design on workers and workplace
- Make explicit the relation of the design to the current and envisioned work tasks

  AS FAR AS POSSIBLE... ... ...

  .....COMPLETENESS NOT POSSIBLE

  BUT...

- Scope of considered tasks and hypothesised effects can and should be made explicit

Work activities amenable to task analysis

Some examples from personal experience:

- HTA — Brick laying, parcel sorting
- TAKD — Programming, word processing, circuit design
- TKS — Architectural design, jewellery design, radiography, explanations, air traffic control, flight information system

Task Knowledge Structures (TKS)

[Johnson 92], [Johnson & Johnson 91a]

- Developed to describe work tasks
- Developed to be used in HCI contexts
- Describes task structure (but not restricted to hierarchical task relations)
- Identifies details of objects and actions
- Design relevance — Adept toolset developed to relate TKS descriptions to design

Task Knowledge Structures (2)

Tasks are not randomly organised:

- The structure in tasks is a reflection of the task, the environment in which it is performed and the experience of the task performer
- This structure is reflected in the person's knowledge of the task and can be identified by a TKS task analysis
- Analysis method: Knowledge Analysis of Tasks [Johnson & Johnson 91]

Task Knowledge Structures (3)

- No assumptions about what is or is not a task
- Task must be defined by workers and work context
- Tasks can involve one or many people
- Tasks can last weeks or just minutes
- Tasks can have relationships to other tasks

5. Introduction to Case Study

- Introduction
- The model-based user interface design paradigm
- Task-based design in Adept
- Task analysis
- Introduction to case study
- Case study:
  - Task model
  - Task design
  - Abstract interaction model
  - Interface generation
  - Design presentations
- Conclusions

The Design Problem

Case study to be undertaken by tutorial participants:

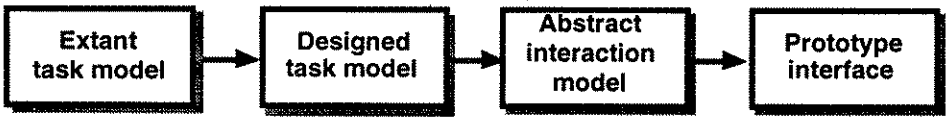
- Design and prototype an interactive system that will support the task of obtaining flight information and booking a flight for a pending trip
- Assume that the travel agent or computerised system has complete information on all flights, routings, pricing etc.
- Focus on the task from the perspective of the traveller (rather than from that of a travel agent or airline)
- Ignore aspects of the task such as consulting diaries, rearranging appointments, taking out travel insurance, etc

Approach

- Perform task analysis and express the results as a task model
- Use task model as the basis for the design of a prototype flight information system using the Adept models and design tools
- Informally assess the differences between your design and others to support the same task

Overview

- Design stages in case study:



- A top-down design approach proceeding from task analysis through to implementation
- Not always appropriate...

6. Case Study

- Introduction
- The model-based user interface design paradigm
- Task-based design in Adept
- Task analysis
- Introduction to case study
- Case study:
  - Task model
  - Task design
  - Abstract interaction model
  - Interface generation
  - Design presentations
- Conclusions

6.1 Task Model

- Variant of Task Knowledge Structures
- Minimal notation facilitating ease of learning and use
- Same notation to express both extant and designed tasks
- Consists of goals, procedures, actions and objects
- Temporal relations defined between goals, procedures and actions

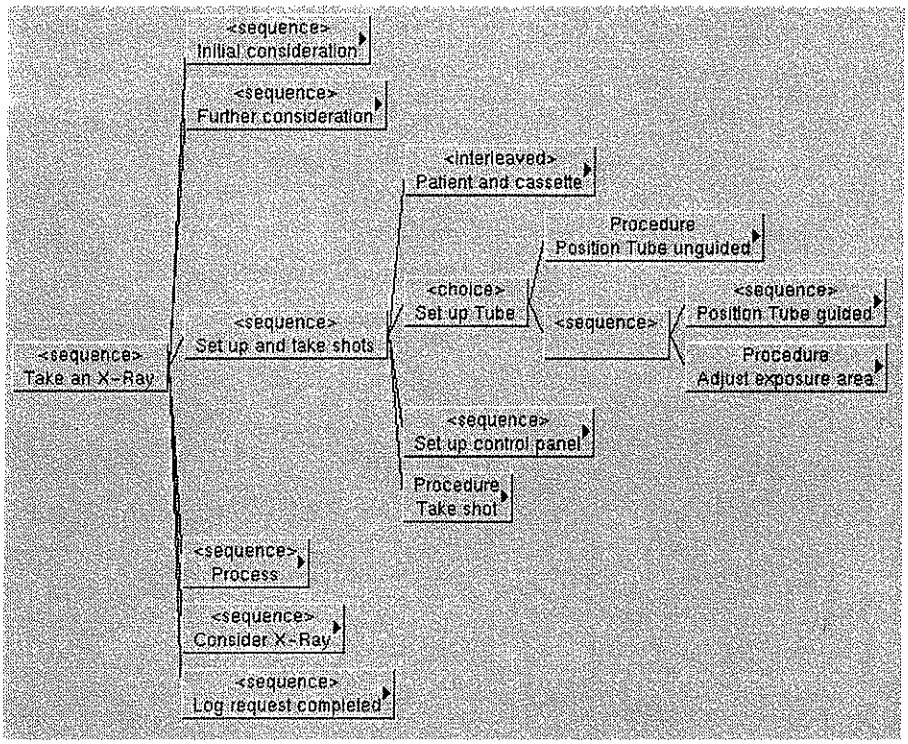
Task Model Components

- *Goal* — the desired state of affairs that a task is directed towards achieving. May consist of a number of subgoals.
- *Action* — the lowest level of activity identified or articulated in the task analysis. Often involves objects.
- *Procedure* — a well-practised sequence of actions (behaviour) directed towards achieving a task goal
- *Object* — an object (information) from the task domain. Defined in terms of attributes, relations to other objects and the actions that may be applied to it.

Temporal Relations

- *choice* — choice between activities
- *sequence* — activities performed in strict temporal order
- *interleaved* — activities performed simultaneously, but only one at any time (multi-threading)
- *parallel* — activities performed concurrently
- *repetition* — activities performed one or more times
- *disable* — one activity interrupts another

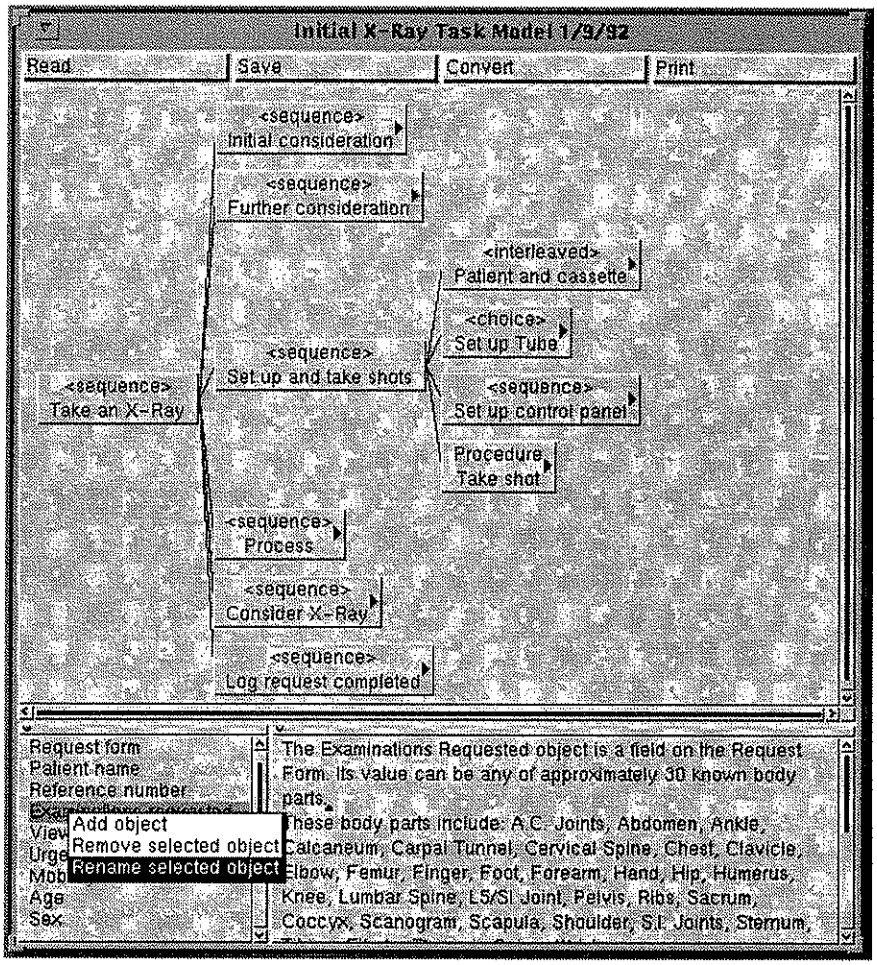
Task Model Example



Adept Task Model Editor (1)

- **Upper window displays a hierarchical view of the task goals, procedures and actions**
- **Displays both task structure and sequencing**
- **Lower windows list the task objects and their descriptions**
- **Structure-directed editing with menu-based interaction**
- **Information hiding**

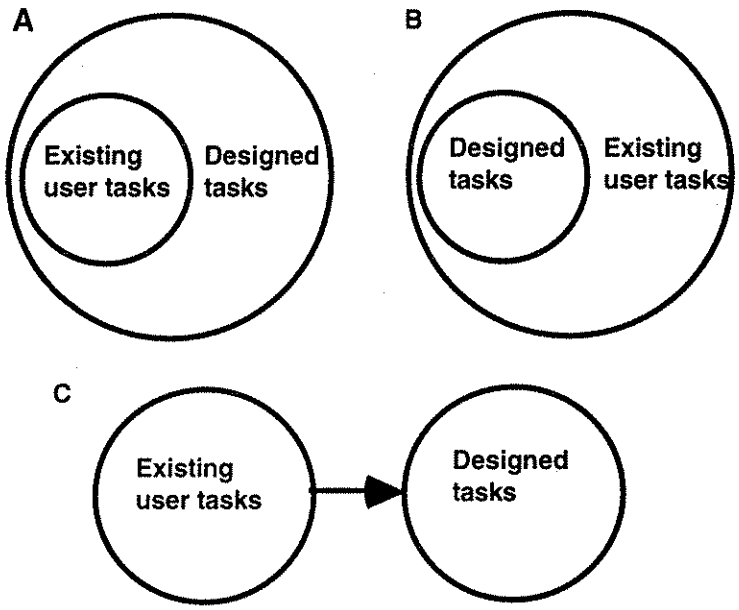
ADEPT Task Model Editor (2)



6.2 Designing a Task

- The designed task model represents the task to be performed with the new artefact
- Created from extant task model by the designer in accordance with design requirements
- Balance between creativity and user centred design
- Explicit nature of designed tasks offers the possibility of early evaluation/exploration of design alternatives

Existing and Designed Tasks (1)



Existing and Designed Tasks (2)

Possible relations between existing tasks and design:

- **Design can extend the range of tasks users can perform (Figure A)**
- **Design can support a subset of existing tasks (Figure B)**
- **Design can replicate current tasks but with a change of technology (Figure C)**

6.3 Abstract Interaction Model (AIM)

- **Represents the transition from designed task to the specification of an interface to support that task**
- **Provides a high-level description of *what* the interface is to do (but not *how* it is to be done)**
- **Models dialogue and interface behaviour**

Features

- Hierarchical structure
- Components:
  - *Groups* — collections of components (either interactors or further groups) with temporal relations to express sequencing constraints
  - *Interactors* — abstractions of interactive behaviour. Categorised in terms of input types, e.g. text, number, selection
- Temporal relations: choice, sequence, interleaved, parallel, repetition, disable

Designing an Abstract Interaction Model

- Simple automated process produces an AIM from a designed task model
- Design activities at the AIM level:
  - refining the design of the artefact including modifications to the temporal relations
  - replacing task actions with interactors
  - iterating between designed task model and AIM

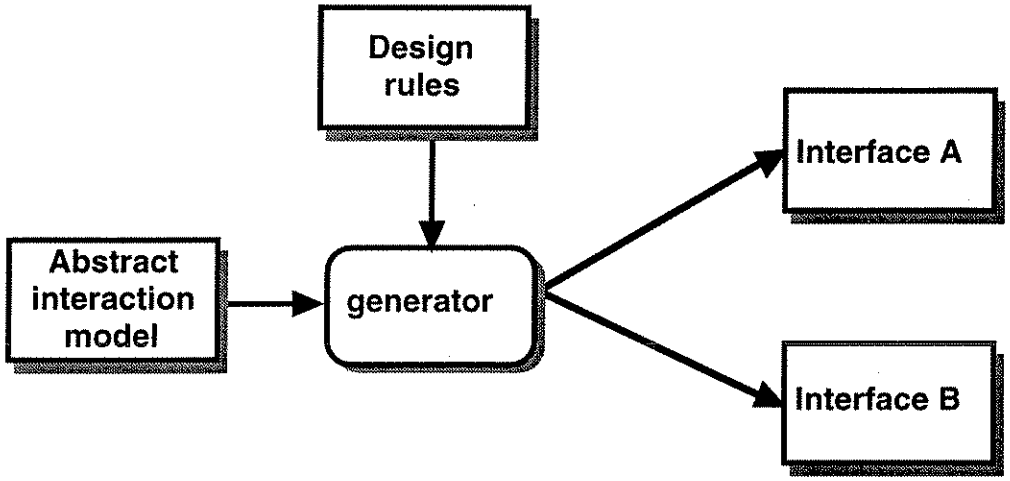
6.4 Prototype Interface

- Also known as the 'Concrete Interface Model'
- A low-level, executable description of the interface design
- Consists of interaction objects (toolkit widgets), with their behaviour, screen layout and sequencing
- May be viewed in either 'build' or 'run' mode, allowing the designer to modify the design and to interact with it

The Interface Generator

- An exploration of intelligent design automation
- Generates user interfaces which satisfy the conceptual level design given in the abstract interaction model
- Actions include:
  - selecting interaction objects
  - generating layouts
  - incorporating sequencing information
- Guided by in-built heuristics and design guidelines from the design rules knowledge base

The Interface Generator



Design Rules

- Knowledge base of design guidelines
- Rules describe mappings between interactors in the abstract interaction model and toolkit components, e.g.

  `numeric interactor —> horizontal slider`

- Rules should encapsulate style and layout guidelines
- Browser tool allows the designer to view and modify design guidelines

6.5 Design Presentations and Discussion

- Benefits of the task analysis?
- Comparisons between participants' designs and other systems (to be described)?
- Problems?

7. Conclusions

- Introduction
- The model-based user interface design paradigm
- Task-based design in Adept
- Task analysis
- Introduction to case study
- Case study:
  - Task model
  - Task design
  - Abstract interaction model
  - Interface generation
  - Design presentations
- Conclusions

Adept Strengths

- Modelling extant and designed tasks in terms of users' work activities
- Representing changes to the users' work through models of designed tasks
- Carrying task information forward into the design of artefacts in a integrated, model-based environment
- Promoting design iteration and artefact evolution
- Demonstrating the application of design automation to well-defined design problems

Adept Weaknesses

- Lack of expressive power in abstract interaction model (especially by comparison with the application models of Humanoid or UIDE)
- Explicit support for top-down design processes only
- Design knowledge restricted to low-level design guidelines
- Lack of argumentation justifying changes to tasks and other models
- Insufficient support for participative evaluation

Recent Extensions to Adept Research

- Undergoing evaluation with interactive system designers with a view to establishing requirements for future development
- Prototype hypertext design rationale facility integrated with all modelling stages
- Prototype model animation tool under development

The Future for Adept ?

- Extending scope of models to include richer descriptions of tasks, interactions and applications
- Enhancing object descriptions
- Supporting bottom-up design by reflecting changes at any level to models at higher levels
- Extending scope of design guidelines
- Migrating models to run-time environment
- Supporting design of multimedia interfaces
- Developing usability evaluation support

Conclusions

- Model-based design focuses on *what* the design is to do, rather than *how* it is to be done
- Task-based design focuses on *why* the design should do what it does
- And the future?
  - integrated design environments
  - design at higher levels of abstraction
  - user centred design
  - intelligent tools

Acknowledgements

Advanced Design Environment for Prototyping with Task Models

- Queen Mary & Westfield College, University of London
- British Aerospace plc, Sowerby Research Centre
- British Maritime Technology Ltd
- MJC^2

Funded by the DTI and SERC (finished July '93)

Research continuing under direct funding from BAe

Bibliography

[Annett et al 71] J. Annett, K.D. Duncan, R.B. Stammers and M.J. Gray . Task Analysis. Training Information Paper No. 6, HMSO, 1971.

[Bodart et al 94] F. Bodart, A-M. Hennebert, J-M. Leheureux, I. Provot and J. Vanderdonckt. A Model-Based Approach to Presentation: A Continuum from Task Analysis to Prototype. In Proceedings Eurographics Workshop on Design, Specification and Verification of Interactive Systems, La Spezia, June 1994, pp. 25-39.

[Card et al 83] S.K. Card, T.P. Moran, and A. Newell. The Psychology of Human-Computer Interaction. Erlbaum, Hillsdale, NJ, 1983.

[Carroll et al 91] J.M. Carroll, W.A. Kellogg and M.B. Rossen. The Task-Artifact Cycle. In J.M. Carroll (ed.), Designing Interaction: Psychology at the Human-Computer Interface, Cambridge University Press, 1991, pp. 74-102.

[de Bruin et al 94] H. de Bruin, P. Bouwman and J. van den Bos. A Task Oriented Methodology for the Development of Interactive Systems as used in DIGIS. To appear in Proceedings 15th Interdisciplinary Workshop on Informatics and Psychology, Interdisciplinary Approaches to System Analysis and Design, Schaerding Austria, May 1994.

[Diaper 89] D. Diaper. Task Analysis for Knowledge Descriptions (TAKD): the method and an example, in D. Diaper, (ed.), Task Analysis for Human-Computer Interaction, Ellis Horwood, 1989, pp. 108-159.

[Foley et al 91] J. Foley, W. Kim, S. Kovacevic and K. Murray. UIDE - An Intelligent User Interface Design Environment. In J. Sullivan and S. Tyler (eds.), Architectures for Intelligent User Interfaces: Elements and Prototypes, Addison-Wesley, 1991.

[Frank & Frank 93] M.R. Frank and J.D. Foley. Model-Based User Interface Design by Example and by Interview. In Proceedings of the ACM Symposium on User Interface Software and Technology, UIST'93, pp. 129-137, 1993.

[Haan 94] G. de Haan. An ETAG based approach to the design of user interfaces. To appear in Proceedings 15th Interdisciplinary Workshop on Informatics and Psychology, Interdisciplinary Approaches to System Analysis and Design, Schaerding Austria, 1994.

[Hartson & Gray 92] H.R. Hartson and P.D. Gray. Temporal Aspects of Tasks in the User Action Notation. Human-Computer interaction, 7, Lawrence Erlbaum Associates, Inc., 1992, pp. 1-45.

[Johnson et al 84] P. Johnson, D. Diaper and J. Long. Tasks, Skill and Knowledge; Task Analysis for Knowledge Based Descriptions. In B. Shackel (ed.), Interact '84, Amsterdam, North-Holland.

[Johnson & Johnson 91a]..H. Johnson and P. Johnson. Task knowledge structures: Psychological basis and integration into system design. Acta Psychologica, 78, 1991, pp. 3-26.

[Johnson 89] P. Johnson. Supporting System Design by Analyzing Current Task Knowledge. In Task Analysis for Human-Computer Interaction, D. Diaper, (ed), Ellis-Horwood, 1989, pp. 160-185.

[Johnson 92] P. Johnson. Human Computer Interaction: Psychology, Task Analysis and Software Engineering, McGraw-Hill, 1992.

[Johnson & Johnson 91b] P. Johnson and H. Johnson. Knowledge Analysis of Tasks: Task Analysis and Specification for Human-computer Systems. In Engineering the Human-Computer Interface, A. Downton (ed), McGraw Hill, London, 1991.

[Johnson et al 93] P. Johnson, S. Wilson, P. Markopoulos and J. Pycock. ADEPT — Advanced Design Environment for Prototyping with Task Models, Demonstration Abstract, Proceedings Interchi '93, ACM, 1993, p. 56.

[Johnson et al 94] P. Johnson, H. Johnson and S. Wilson. Rapid Prototyping of User Interfaces Driven by Task Models. J. Carroll (ed), Scenario-based design, Addison-Wesley (In Press).

[Kelly & Colgan 92] C. Kelly and L. Colgan. User Modelling and User Interface Design. In A. Monk, D. Diaper and M. Harrison, (eds), People and Computers VII, Proceedings of the HCI'92 Conference, Cambridge University Press, September 1992, pp. 227 - 239.

[Kieras & Polson 85] D. Kieras and P.G. Polson. An Approach to the Formal Analysis of User Complexity. International Journal of Man-Machine Studies, 22, 1985, pp. 365-394.

[Kim & Foley 90] W. Kim and J.D. Foley. DON: User Interface Presentation Design Assistant. In Proceedings UIST'90, October 1990, pp. 10-20.

[Lim et al 92] K.Y. Lim, J.B. Long and N. Silcock. Integrating human factors with the Jackson System Development method: an illustrated overview. Ergonomics, 35(12), pp. 1135-1161, 1992.

[Lim & Long 94] K.Y. Lim and J.B. Long. Integration of a Structured Human Factors Method with the Jackson System Development Method. To appear in Proceedings 15th Interdisciplinary Workshop on Informatics and Psychology, Interdisciplinary Approaches to System Analysis and Design, Schaerding Austria, 1994.

[Luo et al 93] P. Luo, P. Szekely and R. Neches. Management of Interface Design in HUMANOID. In Proceedings Interchi'93, April 1993, pp. 107-114.

[Moran 81] T.P. Moran. The Command Language Grammar: A Representation of the User Interface of Interactive Computer Systems. International Journal of Man-Machine Studies, 15, 1981.

[Myers et al 90] B.A. Myers, D.A. Guise, R.B. Dannenberg, B. Vander Zanden, D.S. Kosbie, E. Pervin, A. Mickish and P. Marchal. Comprehensive Support for Graphical Highly-Interactive User Interfaces: The Garnet User Interface Development Environment. IEEE Computer, 23(11), 1990, pp. 71-85.

[Neches et al 93] R. Neches, J.D. Foley, P. Szekely, P. Sukaviriya, P. Luo, S. Kovacevic and S. Hudson. Knowledgeable Development Environments Using Shared Design Models. In Proceedings of the ACM/AAAI International Workshop on Intelligent User Interfaces, January 1993, pp. 63-70.

[Payne & Green 86] S.J. Payne and T.R.G. Green. Task Action Grammars: A Model of the Mental Representation of Task Languages. Human-Computer Interaction, 2, 1986, pp. 93-133.

[Puerta 93] A. Puerta. The Study of Models of Intelligent Interfaces. In Proceedings of the ACM/AAAI International Workshop on Intelligent User Interfaces, 1993, pp. 71-78.

[Puerta & Szekely 94] A.R. Puerta and P. Szekely. Model-Based Interface Development. CHI'94 Tutorial Notes. April 1994.

[Schreiber 94] S. Schreiber. The BOSS-System: Coupling Visual Programming with Model-Based Interface Design. In Eurographics Workshop on Design, Specification and Verification of Interactive Systems, La Spezia, June 1994, pp. 41-59.

[Singh et al 90] G. Singh, C.H. Kok and T.Y. Ngan. Druid: A System for Demonstrational Rapid User Interface Development. In Proceedings UIST'90, 1990, pp. 142-146.

[Siochi & Hartson 89] A.C. Siochi and H.R. Hartson. Task-Oriented Representation of Asynchronous User Interfaces. In Proceedings CHI'89, ACM Press, 1989, pp. 183-188.

[Sukaviriya & Foley 90] P. Sukaviriya and J. Foley. Coupling a UI Framework with Automatic Generation of Context-Sensitive Animated Help. In Proceedings of UIST '90, October 1990, pp. 142-146.

[Sukaviriya et al 93] P.N. Sukaviriya, J.D. Foley and T. Griffith. A Second Generation User Interface Design Environment: The Model and the Runtime Architecture. In Proceedings of Human Factors in Computing Systems, Interchi'93, April 1993, pp. 375-382.

[Szekely et al 92] P. Szekely, P. Luo and R. Neches. Facilitating the Exploration of Interface Design Alternatives: the Humanoid Model of Interface Design. In Proceedings of Human Factors in Computing Systems CHI'92, May 1992, pp. 507-515.

[Szekely et al 93] P. Szekely, P. Luo and R. Neches. Beyond Interface Builders: Model-Based Interface Tools. In Proceedings of Interchi '93, April 1993, pp. 383-390.

[Tauber 90] M. Tauber. ETAG: Extended Task Action Grammar — A Language for the Description of the User's Task Language. Proceedings of INTERACT'90, D. Diaper, D. Gilmore, G. Cockton and B. Shackel (eds.), North-Holland, 1990, pp. 163-174.

[Vanderdonckt & Bodart 93] J.M. Vanderdonckt and F. Bodart. Encapsulating Knowledge for Intelligent Automatic Interaction Objects Selection. In Proceedings of Interchi '93, April 1993, pp. 424-429.

[Walsh et al 88] P.A. Walsh, K.Y. Lim, J.B. Long, and M.K. Carver. Integrating human factors with system development, in Heaton, N. and Sinclair, M. (eds.), Designing end-user interfaces, Oxford: Pergamon Infotech, 1988.

[Wiecha et al 90] C. Wiecha, W. Bennett, S. Boies, J. Gould and S. Greene. ITS: A Tool for Rapidly Developing Interactive Applications. ACM Transactions on Information Systems, 8(3), July 1990, pp. 204-236.

[Wilson et al 93] S. Wilson, P. Johnson, C. Kelly, J. Cunningham and P. Markopoulos. Beyond hacking: a model based approach to user interface design. In Proceedings of HCI'93, J. Alty, D. Diaper and S. Guest (eds), Cambridge University Press, 1993, pp. 217- 231.

Appendix: Details of Radiography Case Study

X-rays are used for a variety of purposes from diagnosis to treatment. This case study involved a radiography unit producing X-rays for use in clinical diagnosis. The task is performed by radiographers and the X-rays are later passed to radiologists who carry out the specialist interpretation. Under normal circumstances, only one radiographer will be involved in taking an X-ray.

An Extract from a Radiographer's Task Scenario

Taking an X-ray examination in normal circumstances involves:

- Receiving a request either on a paper form from a hospital casualty doctor, ward doctor, GP, outpatients department etc. or as a computerised request from wards within the hospital (examples of these are provided to the designer). Previous X-rays may accompany the request form.

- On the basis of this information, deciding which X-ray room to use, which camera to use, which position to put the patient in (standing, on a fixed bed, seated), which area of the body to focus on, where to put the film cassette (under the limb, under the bed etc), whether or not to use a contrast grid, and which film type and size to use.

- Collecting the patient, positioning them, using extra protective aprons or foam positioning pads, and comforting the patient while giving them information about what is happening and instructions about what they must do.

- Preparing the camera by unlocking it, moving it in various planes, locking it, turning it on and focusing it.

- Rechecking the patient's position.

- Leaving the patient and moving behind a protective screen to set the controls for taking the X-ray examination.

- Setting up the control panel. This involves deciding the current ('mA'), the voltage ('kV') and the time ('mS') for the exposure. These choices can be made from memory, by using a wall chart, or by using pre-set buttons on one of the control panels. A number of other settings can also be made (e.g. choice of tubes, choice of focus of the beam).

- Taking the X-ray. There is visual and auditory feedback when this happens.

- Removing the X-ray plate.

- Stamping the X-ray with the patient's name.

- Placing the film in a machine to be developed and, while waiting, reloading the film cassette using one of four machines (according to film type and size).

- Stamping the developed X-ray with the date, examining it on a lightbox, marking it with extra information for the radiologist using a pen (e.g. left and right, standing, etc),and giving it a coloured label according to how many other X-rays the patient has had at this hospital.

- Performing some preliminary interpretation to decide if the X-ray that has been taken will provide the radiologist with the information required.

- Potentially rejecting an X-ray if, for example, it is over-exposed and too dark.

- If an X-ray is rejected, completing a 'reject analysis' form and stating what type of X-ray it was, which room was used, the size of film, and the cause of rejection (e.g. patient moved, over exposure, etc).

- Possibly deciding to take another X-ray, for example, if one has been rejected, or a fracture is longer than thought and continues beyond the edge of the X-ray.

- Taking further X-rays may involve the resetting of the machine (often manually). Also, X-rays are often taken in pairs at right angles to each other.

- When satisfied with the X-rays, returning the patient (often with their X-rays).

- Finally, logging that the request has been completed. If the request came in as a paper form, then all the details must be entered into the computer. In this case, the paper request forms may be passed on to entry clerks. If the request was received via the computer then the patient's reference number is used to record that an X-ray has been taken.

An Extract of a Scenario Featuring Radiographers' Current Equipment

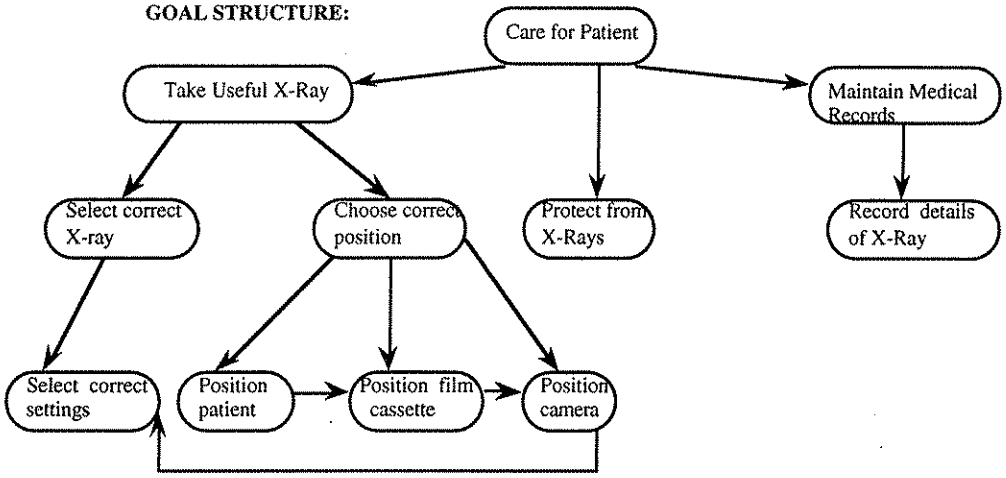
Radiographers categorise X-rays in terms of body parts. The request forms that come to the radiographers name the body part for which an X-ray is needed. From the choice of the body part there follows knowledge of the standard exposure, the standard views (i.e. the positions of the camera and patient), required film size etc.

At present, the radiographers use two X-ray rooms with different control panels. The older room uses manual controls and has a wall chart that lists body parts together with their appropriate exposure settings. The newer room has a more modern control panel with pre-set buttons built into the display. The pre-sets are arranged in rows and the rows appear next to a drawing of a body. This drawing of a body does not do anything. It looks as if the rows of pre-sets should correspond to the general area of the body at the same heights - e.g. the first row might be skull, jaw, neck, etc. Some of the pre-sets do seem to roughly correspond to the general area of the body at the corresponding height, but many do not and are allocated in a seemingly random manner. Consequently, the spatial layout and the categorisation of pre-sets are inconsistent and do not make much sense to the users (they often have to flip through all the rows to find the body part that they are seeking).

Example TKS for an X-ray Task

TASK: Taking an X-Ray
ROLE: Radiographer
CONTEXT: Casualty X-Ray Theatre
ORGANISATION: London Hospital

GOAL STRUCTURE:



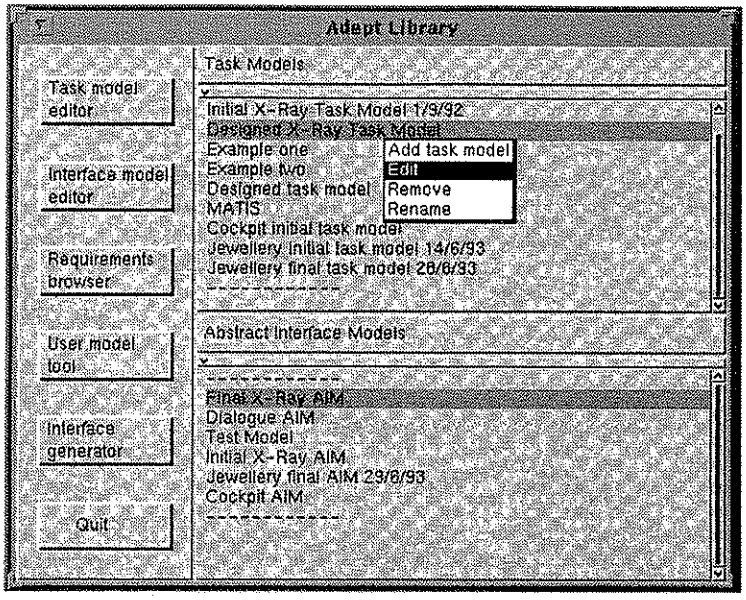
PROCEDURES

```
Check height of camera
  If camera not at correct height then adjust camera
  If camera at correct height then check horizontal  position of camera
    Adjust camera
      Unlock height  adjuster
      If camera too high then raise camera Until camera at correct height
      If camera  too low then lower camera Until camera at correct height
      Lock height adjuster
Check horizontal position of camera
  :
  :
```

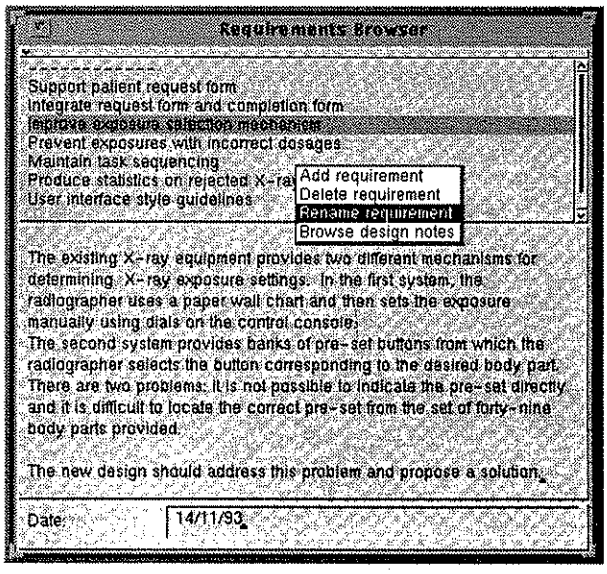
OBJECTS:

OBJECT LIST	OBJECT PROPERTIES	**TYPICAL**	**CENTRAL**
Request Form →	Is a Medical Record →	Very Typical →	Crucial
X-Ray record	Has a Patient Name	Best example	Very important √
X-Ray film	Has an Admission	is :	Important
Body Parts	Has a Patient History	Request Form	Not very important
	Is made of paper	type XRT1	Optional
	Is used in X-ray task		
	Is used by radiographers		
	Is used by clerical staff		
	Is used by nursing staff		
	Is used by clinical staff		
	::::::::		
	::::::::::		

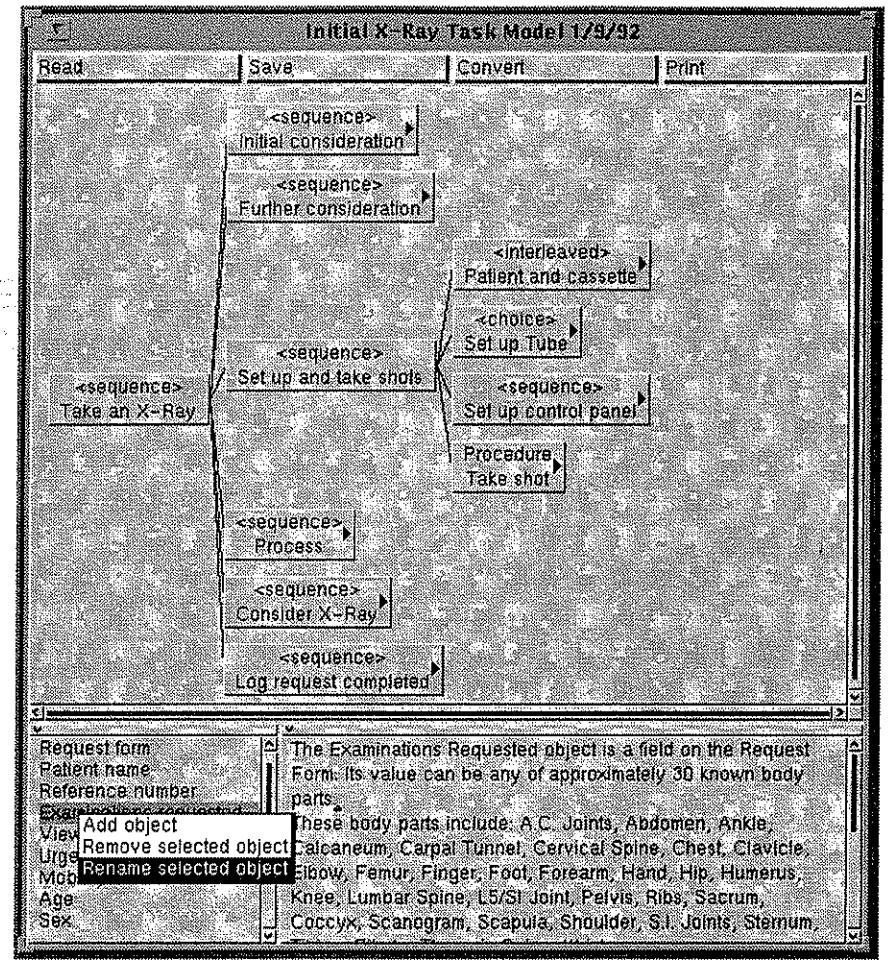
Adept Model Library



Adept Requirements Browser

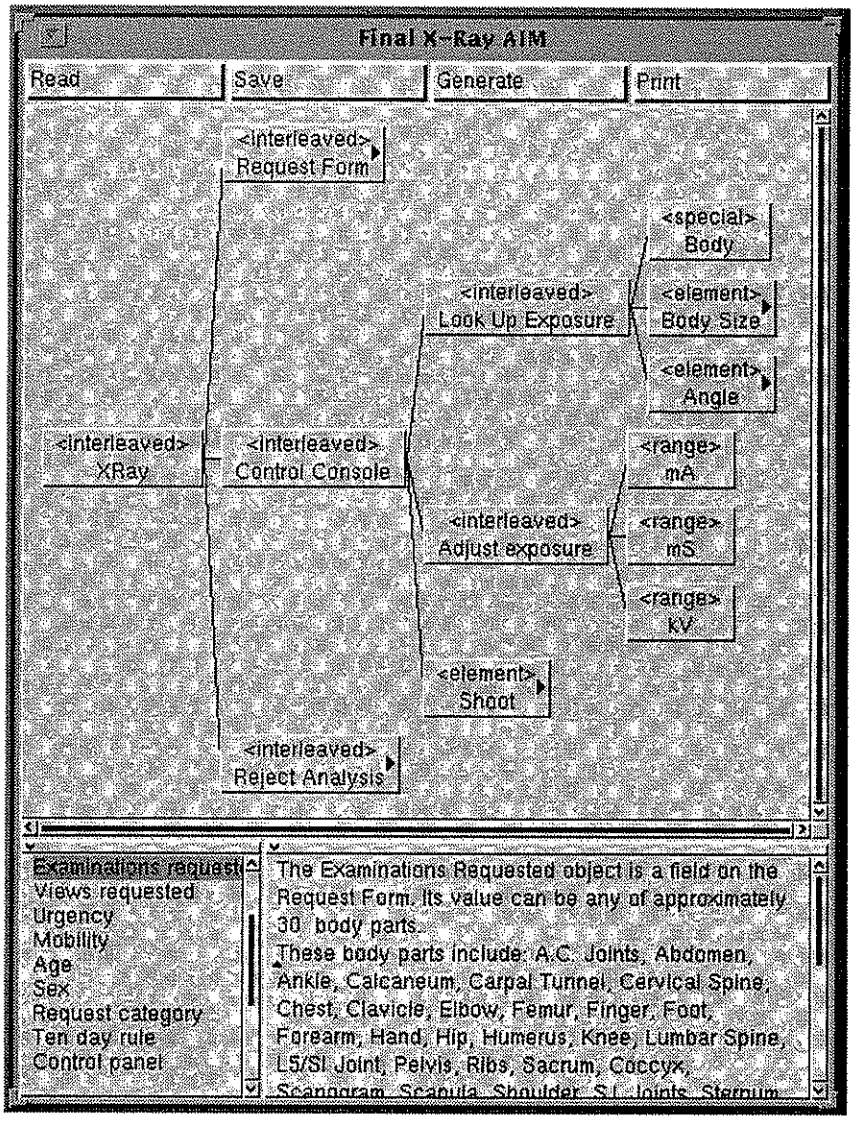


X-ray Extant Task Model



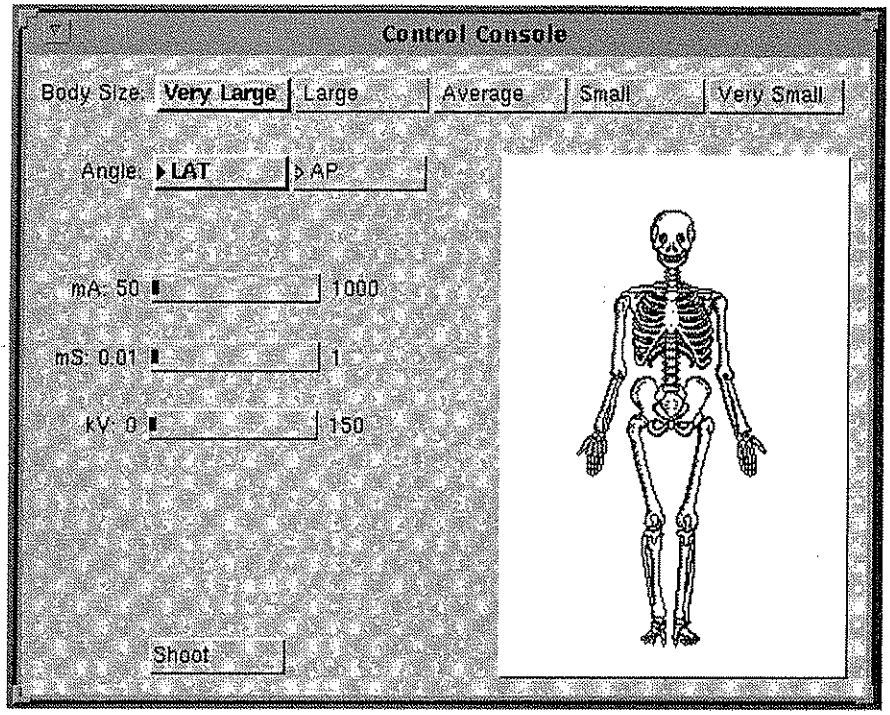
Top-level of X-ray Extant Task Model

X-ray Abstract Interaction Model



Top-level of X-ray Abstract Interaction Model

Prototype Interface



One window from the prototype interface generated to support the X-ray task