Task Model definition and Task Analysis process

Cuno Duursma
Vrije Universiteit Brussel
AI Laboratory
Pleinlaan 2
B-1050 Brussels
email: cuno@arti.vub.ac.be

Document Id.	KADS-II/M5/VUB/RR/004/1.1c
CEC Reference	Deliverable M.5, Version 1.1c
Date	December 15, 1993
Distribution	Consortium

KADS-II Consortium

This document is part of a research project partially funded by the ESPRIT Programme of the Commission of the European Communities as project number 5248. The partners in this project are Gap Gemini Innovation (F), Cap Gemini Programator (S), Netherlands Energy Research Foundation ECN (NL), ENTEL SA (ESP), IBM France (F), Lloyd's Register (UK), Swedisch Institute of Computer Science (S), Siemens AG (D), Touche Ross MC (UK), University of Amsterdam (NL) and Free University of Brussels (B).

Cap Gemini Innovation
Paris Research Centre
86/90, rue Thiers
92513 - Boulogne
France
Contactperson: M. Georges
Phone +33.1.49.10.53.98 Fax: 33.1.49.10.06.15
E-mail: mari@capsogeti.fr

Cap Gemini Programator
Norra Ringv. 36 Box 7
S-721 03 Västerås
Sweden
Phone: +46 21 14 44 50 ; Fax: +46 21 11 41 00
Contactperson: W. Menezes
E-mail: wm@industri.vs.cgl.se

Netherlands Energy Research Foundation ECN
Software Engineering & Research Department
P.O. Box 1, NL 1755 ZG Petten
The Netherlands
Phone: +31 22 46 44 58; Fax: +31 22 46 18 64
Contactperson: Dr. J.M. Akkermans
E-mail: akkermans@ecn.nl

ENTEL SA
P. de la Castellana 141
28046 Madrid
Spain
Phone: +34 1 348 12 23 ; Fax: +34 1 279 10 74
Contactperson: S. Gala
E-mail: sgala@ic.eritel.es

IBM France
Rue de la vieille Poste
BP 1021, 34006 Montpellier
France
Phone: +33 67 34 40 43; Fax: +33 67 34 48 96
Contact person: G.Rondot
E-mail: rondot@mopprofs.vnet.ibm.com

Lloyd's Register
29, Wellesley Road
Croydon CRO 2AJ, Surrey
United Kingdom
Phone: +44 816 814 040 ; Fax: +44 816 816 814
Contactperson: R. Martil
E-mail: rob@lloydreg.uucp

Swedish Institute for Computer Science
KBS Department
Isafjordsgatan, 16428 Kista
Sweden
Phone: +46 87 52 15 20 ; Fax: 46 87 51 72 30
Contactperson: R. Gustavsson
E-mail: rune@sics.se

Siemens AG ZFE BT SE 1
Otto Hahn Ring 6
D-81739 München
Germany
Phone: +49 89 636 40798 ; Fax: 49 89 636 42284
Contactperson: C. Bauer
E-mail: cb@zfe.siemens.de

Touche Ross Management Consultants
The Knowledge Based Systems Centre
Peterborough Court, 133 Fleet Street
London EC4A 2TR, United Kingdom
Phone: +44 71 936 3000 ; Fax: +44 71 583 1198
Contactperson: L. Land
E-mail:

University of Amsterdam
Department of Social Science Informatics
Roetersstraat 15, NL-1018 WB Amsterdam
The Netherlands
Tel: +31 20 525 6789; Fax: +31 20 525 6896
Contactperson: B.J. Wielinga
Electronic mail: wielinga@swi.psy.uva.nl

Free University of Brussels
Artificial Intelligence Laboratory
Pleinlaan 2, B-1050 Brussels
Belgium
Phone: +32 2 641 641 3700 ; Fax: +32 2 641 3729
Contactperson: W. Van de Velde
E-mail: walter@arti.vub.ac.be

This document was produced using the latex wordprocessor

Task Model definition and Task Analysis process

Cuno Duursma
Vrije Universiteit Brussel
AI Laboratory
Pleinlaan 2
B-1050 Brussels
email: cuno@arti.vub.ac.be

Synopsis

This document is version **1.1c** of deliverable M.5, Task Model Definition and Task Modelling process. It describes the definition, function, notation and management of the task model. It also provides an elaborate case study in which these methodological tools are applied. This is a living document, intended to be further improved during the project into Version 2, driven by validation, review activities, identification of scenarios and approaches.

Document Id. KADS-II/M5/VUB/RR/004/1.1c
CEC Reference Deliverable M.5, Version 1.1c
Date December 15, 1993
Distribution Consortium

Version 1

Document history

Author: Cuno Duursma

Revision date: February 1993

Modifications: Version 0.8

- Collection of Task Model information from older versions and Case example from T1.2.3 [Duursma et al., 1992].

Author: Cuno Duursma

Revision date: May 1993

Modifications: Version 1.0

- First official version. Deliverable M5 Version 1.
- Comments of internal reviewers processed.

Author: Cuno Duursma

Revision date: June 1993

Modifications: Version 1.1

- Comments of external reviewers processed.
- Updated with results of KADS plenary.
- Model Coherence improved.
- Model slots and relations better described and made consistent with the entity relation diagram.
- Format of risk management guidelines improved.
- Various suggestions for improvements from project members included.

Author: Cuno Duursma

Revision date: June 1993

Modifications: Version 1.1b

- Writing style improved.
- Model relations defined by 'between' instead of 'from-to'.
- Extensional/intensional development-state-definitions replaced by sufficient/necessary conditions.

Author: Cuno Duursma

Revision date: December 1993

Modifications: Version 1.1c

- Errors in arrows of entity relationship diagram solved.
- Some updates to Task Model relations.

To do

- Validate the links to the other models. The 'direction' of the relations (arrows in the E/R diagram) is still inconsistent here and there.
- Develop more 'guidelines'.
- Validate the 'modelling states'.
- Extend the example.
- Links to other models need to be checked by the developers of the other models.
- Develop "guidelines for the selection of the correct type of 'leaf tasks"' to allow for a smooth transition to expertise modelling.

Contents

Chapter 1

Model Definition

1.1 Introduction

This document presents one of the models of the *Common*Kads model set: the task model (TM). This document has been based on earlier work on task modelling, within KADS [de Greef and Breuker, 1990] and outside KADS [Rademakers, 1991, Steels, 1990, Mcgraw and Herbison-Briggs, 1989]. The innovation of the task model in *Common*Kads consists of (1) the use of several versions of the task model in order to model current, intermediate and required situations, and (2) the embedding of the task model in a risk driven model development.

The 'task model definition' is a template to describe functions of an organisation in terms of tasks. Task *analysis* helps the knowledge engineer to organise a view on an expert's major tasks in a given area and helps to set the scope for the KBS solution, supports project feasibility analysis and much more.

A task statement refers to a 'coherent set of activities'. For the task model the actual activity is not important, but the intention or purpose of the set of activities.

A task model is an engineering artefact, designed by the knowledge engineer, which does not necessarily correspond to a cognitive model of the domain expert's problem solving behaviour [Rademakers, 1991].

Several task models can be instantiated at different stages of a project. An initial task model describes an organisation as it was found at the start of project. A required task model is the result of an organisational analysis and investigating the application of automation techniques. A 'current' task model could describe the tasks at a certain stage of the project: where some changes in the organisation have taken place and some systems have been implemented and deployed within the organisation.

The task model is a template, based on our experience with relevant features of the task analysis process. Guidance about task analysis is represented in *Common*Kads by *modelling guidelines*, listed in [Aamodt et al., 1992]. These guidelines provide explanation and examples of activities and reasons why they can be considered a task and what attributes or relations should be represented. Guidance about 'when' and 'what kind of' task model should be developed is formulated by *risk management guidelines*. Some risk managements guidelines are formulated in Sec. 1.5.2.

1.2 What is a task?

A task statement refers to a set of coherent activities that are performed to achieve a goal in a given domain. Here 'activity' refers to actual actions that are performed in the world, whereas a task statement adds a teleological dimension to these actions. For example the activity of hitting a nail with a hammer may correspond to the task of sinking the nail in the wood. It may also correspond to the task of making a noise to attract someone's attention.

It is important to note that in all practical contexts where task modeling is being done, one must make an application-specific decision on what the granularity of activities will be. If activities are too fine-grained, the size of the task model might be too big for practical validation.

It is not possible to define the concept of task in terms of necessary and sufficient conditions. But we are able to describe what kind of activities and behaviour can be considered tasks and applied in a useful way in the methodology (adapted from [Mcgraw and Herbison-Briggs, 1989]):

1. A task has a definite beginning and end. It is performed in a relatively short period of time.
2. A task statement describes a finite, independent part of the job.
3. Each task statement should be clearly understood by anyone doing the job. Its description should use terminology that is consistent with current usage by the target group.
4. Each task should be tractable in terms of time spent doing it.
5. A task statement should elicit differences among respondents; there should be those who do it and those who do not.
6. A task begins and ends by an observable clues.
7. A task statement is brief.
8. A task statement avoids qualifiers. Only when there is more then one way to perform a task, is a qualifier is allowed.
9. A task statement uses one verb, except when several actions are invariably performed together.
10. A task must be measurable. The end result or product can be estimated or measured as having been done correctly/successfully.

Task models are subjective models. This subjectivity is twofold: (I) the observer has to make biassed observations and (II) there is a bias inherent in the resulting abstractions that are being produced [Rademakers, 1991]. Activities can thus be grouped into different tasks, depending on the role of the task model in the development process of the KBS.

1.3 The role of task models

The task model stores the results of the task analysis process. In this section we will try to answer the question: why it is useful to make a task analysis and to instantiate a task model.

Task analysis is performed in order to find out several things about the activities of people in an organisation. According to [Mcgraw and Herbison-Briggs, 1989] quoting [Shannon, 1980], task analysis is a methodological tool that can be used to (1) describe the functions a human expert performs, and (2) determine the relation of each task on a certain dimension to the overall job. Task analysis is used to find out what a design requires as well as what design is required by Human factor engineers. Task analysis is also used to establish requirements for training and for user documentation.

No formal models for a 'good' task analysis are currently available. Consequently there are no established rules to turn task analysis into a routine activity and *function* should therefore determine *form* [Woodson, 1981]. We use a generic form, a template that can be specialised for its specific use, which will be introduced in the next section. The generic form needs to be instantiated for a specific task analysis; stating which of the 'options' will be used in the task model to document the task analysis.

Task analysis makes the purpose of activities that are performed or need to be performed within the organisation explicit. This is done from an abstract level down to rather basic levels. This is needed to get an overview of what is actually happening (analysis) and to plan what should be done (design). However, the process of analysis and design is iterative. Thus, the initial result should be considered preliminary in nature.

Task models are documentation structures that are used in *Common*Kads for:

- Documenting the result of an analysis of the current activities.
- Documenting the result of a task design of proposed activities.
- Supporting organisation change management.
- Setting the scope of the KBS solution.
- Supporting project feasibility assessment.
- Supporting personnel selection.
- Identifying needs for Training.
- Assessing efficiency, robustness and quality of work in the organisation.

The function of the task model template is to support the representation and structuring of selected pieces of information. Relevant information includes the type of the input and output, the requirements of the input-output relation, the frequency of the task performance and the relation with other tasks. When this information has been inserted, the model can be used to support the activities listed above. The task model does not make predictions about the results of these activities, but allows information to be structured so that it is easier to perform them.

1.4 The contents of the task model

1.4.1 Aspects of tasks

Aspects of tasks we are interested include: the name, a short description of the purpose and activities involved, input and output ingredients, the goal of the task and its functions or purpose in the task structure, its control, frequency of application, its super and sub-tasks, and its agent(s) or distribution. For the various applications of *Common*Kads it can also be useful to include:

- The representation of both the goal of the task and the degree to which the goal is generally achieved. For instance, the goal of the task might be to diagnose all the faults of cars to the level of the smallest repairable components available, while the mechanic succeeds in diagnosing only 85% of the cars in that way, and repairs the remaining cars by replacing larger, more costly components.
- The association between the task-model representing the current situation and the task model representing the required situation. For instance, the task structure can be made of the situation before a KBS has been introduced, and one can be made after completion and installation of the system. Links can be made to show how certain goals are redistributed and reassigned in the new structure.
- The preconditions of a task, either in terms of the accomplishment of other tasks or the availability of input ingredients can be represented. For instance, in order to start diagnosis of a car, the car should have been moved into the garage.
- The control over the sub-tasks.
- A description of required knowledge or skills needed to complete a task. If the task is assigned to the KBS agent then the required knowledge should be included in an expertise model. If the task is assigned to other agents then an abstract description of the skills required could be represented in a task model.
- The representation of different categories of tasks, like time categories (a set of task performed at noon), communication categories (task that use similar input), functional categories (tasks that contribute to the same function), and organisational categories (tasks applied by a certain agent or department in the organisation).
- The type of task decomposition. Some tasks decompose along the dimension of time, others along the type of ingredients involved, or along independent sub-goals. Identifying the type of decomposition can increase the understanding of the task structure and supports the consideration of alternative decompositions.

1.4.2 Task model template

The *Common*Kads task model template contains by default the following slots and relations[1].

[1] *obl* refers to a slot/relation that has to be instantiated for any task identified, *Opt.* refers to optional slots/relations, *1, single, many* refers to the number of allowed instantiations

The following entity types are defined in the task model:

Task *entity* Representation of a task. Tasks are in the center of the task model. All other entities owe their existence to a task, in the sense that they cannot exist unless they are related to some task entity.

input *relation, (Obl.) 1 to many, between task and ingredient.* 'Takes' describes the type of inputs that is used by the task. Locally used information is not considered input. Additionally it can be specified where the input comes from and the format of the input.

output *relation, (Obl.) 1 to many between task and ingredient.* The information that is released by the task. It can also be specified where output goes to. Feedback can be modelled in this way.

goal *slot, (Obl.) 1 to many.* These are requirements onto transforming input into output.

description *slot, (Obl.) 1 .* A description of the task in natural language.

precondition *slot, (Obl.) 1 to many.* The precondition(s) of a task instance represent(s) requirements on the environment, apart from availability of input, that should hold before it can be executed.

control structure *slot, (Opt.) 1.* Control over the sub-tasks and over activities for the leaf-tasks.

Performance time *slot, (Opt.) 1.* Typical or permissible completion time (task analysis) or allowed completion time (task design).

Performance *slot, (Opt.) 1.* Typical or permissible quality of the actual output in relation to the goal norm.

frequency pattern *slot, (Opt.) 1.* How often a task is performed. This is also control, but from the task perspective itself. Frequency can be absolute (e.g. once a week), relative to the input (e.g. for every item in the input), or relative to the frequency of super tasks. It can also be different for the various outputs; e.g. one output every week and another one every day.

super task *relation, (Opt.) 1 to many, between two tasks.* The task this task is a sub-task of. The goal of the task is always a sub-goal of its super-task.

sub-task *relation, (Opt.) 1 to many, between two tasks.* The sub activities of a task to perform the goal. The sub-tasks always satisfy sub-goals of this task. Their frequency can depend on the frequency of this task.

feature *relation, (Opt.) 1 to many, between task and feature.* The features are a characterise a task in terms of an abstract language (feature grammar). This language is interpreted by the model adviser tool but should also make sense to the knowledge engineer. Task features are described in [Aamodt et al., 1992].

type *relation, (Opt.) 1 to many, between two tasks.* The type 'relation' provides a way of modelling task taxonomies.

requires *relation, (Opt.) 1 to many, between task and capability.* The skills required of an agent to perform this task successfully.

decomposition type *slot, (Opt.) 1* The type of decomposition relation between a task and its sub-tasks. This slot provides a rationale for the task decomposition of this task. There may be several different ways of decomposing a task, and by providing this 'decomposition type' it is easier to understand why the specific subtasks were identified and why the specific control structure was specified.

Feature *entity* Representation of a task feature. Task features characterise a task in terms of an abstract language (feature grammar). This language is interpreted by the model adviser tool but should also make sense to the knowledge engineer. Task features are described in [Aamodt et al., 1992].

name *slot, text, (Obl.), 1.* The name of the feature is a short text string.

category , *slot, text, (Obl.), 1.* This is the classification of the feature, according to the standard task feature categories.

property , *slot, text, (Obl.), 1.* This is the general property described by this feature.

value , *slot, text, (Obl.), 1.* This is the value expressing the feature property of the related task.

Ingredient *entity* Representation of a task input or output ingredient. An ingredient describes the contents of information being used or produced by tasks. This description is oriented towards the semantics of the information, not how it is represented syntactically.

name *slot, text, (Obl.), 1.* The name of the ingredient is a short text string. The name must be unique across the ingredients of the task model, as it is the main reference mechanism used for specifying relations with ingredients.

contents *slot, text, (Obl.), 1.* This slot provides for the description of the contents of the ingredient. The level of detail in this description is dependent on the criticality of the ingredient, from the point of view of understanding what the task needs to handle, and of understanding the requirements on communication involving this ingredient.

Capability *entity* Representation of a capability needed for performing a task. As there are more detailed representations of capabilities available in the agent and communication models, the capability entities in the task model are intended to represent a more high-level view of the competence needed for task performance.

name *slot, text, (Obl.), 1.* The name of the capability, used for cross-referencing in the task model.

description *slot, text, (Obl.), 1.* An informal description of the capability requirements.

Environment *entity* The environment entity represents those formal constraints on task performance, that are established within the organisation, through general laws, by professional or trade rules, etc.

name *slot, text, (Obl.), 1.* The name of the environment, used for cross-referencing within the task model.

norms *slot, text, (Opt.), 1 to many.* A description of the norms that specify good practice in performance of the task.

constraints *slot, text, (Opt.), 1 to many.* A description of the constraints that may not be broken. Examples of such constraints are laws and security rules.

systems *slot, text, (Opt.), 1 to many.* A description of the systems that exist in the environment, and that provide a set of constraints on how tasks can be performed.

Fig. 1.1 is an entity-relationship-diagram of the task model. It puts the task model concept in the centre, with its relations to other entities and the five other models of the *Common*Kads methodology.

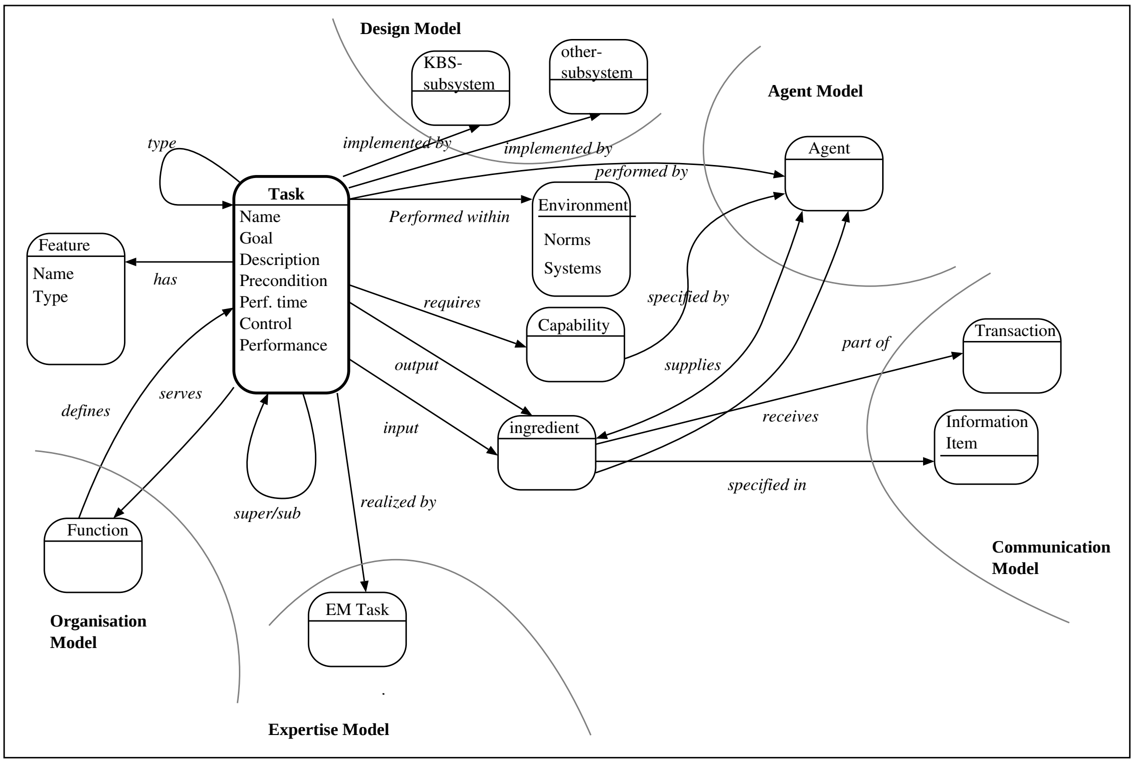

Figure 1.1: Entity-relationship-diagram of the task model of the *Common*Kads methodology.

1.4.3 External dependencies

The task models have dependencies with the organisational model, with the expertise model, communication model, agent model and design model. The dependencies are shown in detail in Fig. 1.1, any relation between the task model and another model is an 'external dependency'.

The information contained in phases in traditional software engineering like 'requirements specification' or 'design' [Sommerville, 1989], is distributed over different versions of the five specialised models in *Common*Kads (see [Duursma et al., 1992]). These models should be consistent. For instance, the task model should comply with the organisational model and the expertise model, with the communication model. These models are developed independently from each other. The requirements of each model depend on the requirements on one or more other models. In this section we investigate the links between the task model and the other models it has relations with, as shown in Fig. 1.1.

Organisation model The Organisation model provides a rationale, from a business point of view, for performing task modeling on some tasks identified in the organisation.

If the Organisation model has been developed, and a function has been identified for detailed elaboration in the task model, then there will be links established between that function and a top-level task in the task model.

If an initial decision was made not to develop an Organisational model, there will of course be no links between the Task and Organisational models.

serves *relation, (Opt.) 1 to many, between task and function.* A task serves a function in the Organisation Model.

defines *relation, (Opt.) 1 to many, between function and task.* The task model only contains only those tasks which are directly related to the realisation of the function of the organisation.

Agent model The Agent model provides a framework for elaborating on the competences needed by agents performing tasks, or providing/receiving information used by tasks.

performed-by *relation, (Opt.), 1, between task and agent.* Tasks are assigned to agents represented in the agent model.

supplies *relation, (Opt.) 1, between agent and ingredient.* Ingredients of leaf tasks are supplied by agents.

receives *relation, (Opt.) 1, between ingredient and agent.* Ingredients of leaf tasks are received by agents.

Communication model The communication model provides a framework for elaborating on the structure and contents of task-agent communication.

receives *relation (Opt.) 1, between transaction and ingredient.* Every 'ingredient transfer' between leaf tasks is described as a transaction in a communication model.

specified-in *relation (Opt.) 1, between ingredient and information item.* Every 'ingredient' is specified as an information item in a communication model.

Expertise Model A complex task can be modelled in detail in the expertise model.

realised-by *relation, (Opt.) 1 to many, between task and expertise model task (task header).* A task is realised by a task in the expertise model.

Design Model For tasks that will be performed by computer systems, the design model provides the operationalisation of the task.

implemented-by *relation, (Opt.) 1 to many, between task and design model* KBS *subsystem or design model other subsystem.* A task is implemented by either a KBS subsystem or an 'other' subsystem in the design model.

1.4.4 Task representation and notation

The task model is a model with little formal structure. The notation can therefore be left quite open; the user can either take the notation provided in this report and the case study, or adapt the visualisation to a familiar standard.

Approaches to task modelling will be discussed in detail in the next version of this deliverable. Notations that will be useful include structured text, graphes, tables and also formal representations.

1.4.4.1 Structured text

The results of a task analysis are usually documented as structured text. The structure guides the reader through the text by a uniform organisation. This organisation can take the form of frames, graphes, Venn diagrams, hyper-text etc. In *Common*Kads the task concepts are represented by frames, and the task/sub-task relation is represented by a graph or a tree (see Fig. 1.2). Other task relations like task classes are usually not drawn. A task is taken as an entity and the slots of the task templates are filled by texts in natural language.

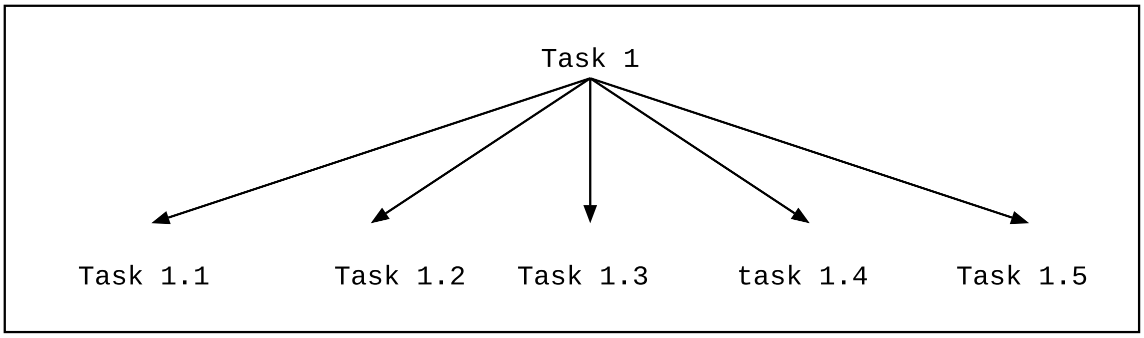

Figure 1.2: A notation template to visualise the sub-task attribute of a task

Other formats can be used, for instance by showing dependencies and duration for time-line analysis, also used in project planning systems (see Fig. 1.3). The required tasks shown by horizontal bars that represent the duration of the task, and the tasks that create the (required) input are shown to end before this bar, as in Fig. 1.3. Various specialisations of this layout can be made, to show optional and required inputs, real time constrains, parallelism etc.

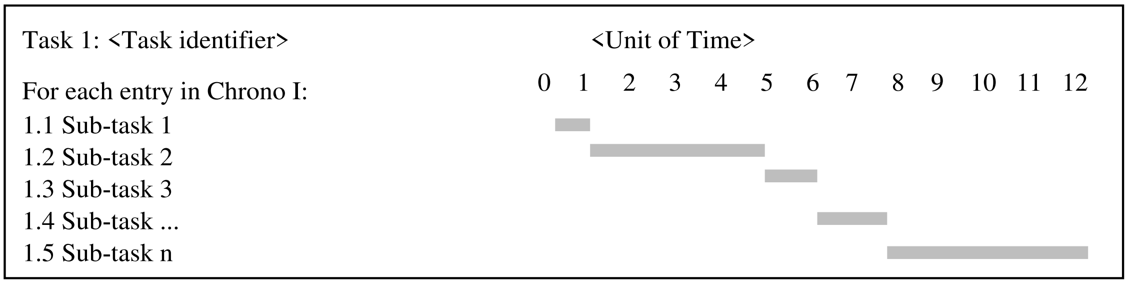

Figure 1.3: A notation template to show duration and precondition attributes of tasks.

1.4.4.2 Formal task representation

A formal representation of a task is obviously not only a notation, but a representation of a task in terms of a set of symbols that are manipulated according to specific rules (the syntax), and have a well described interpretation (the semantics). It can indeed be useful to formalise certain attributes and relations of the leaves of the task decomposition hierarchy. Attributes that can be formalised are input, output, goal, sub and super task.

Formalised leaf tasks can be compared with formal requirement definitions in software engineering [Sommerville, 1989]. In the *Common*Kads task model we do not provide contributions to formal specifications of requirements formulation. *Common*Kads users can select from a range of existing techniques for this purpose. Formalising the requirements can be a good idea in certain circumstances, depending on the risks involved.

The risk driven approach guides this choice. For instance, if the requirements to a task can be specified very accurately and needs to be strictly adhered to, then there is the risk that the task model will not completely satisfy this requirement. This risk can be reduced by formalising the requirements by using formalised input, output and goal attributes, and showing that, for instance, the decomposition into sub-tasks satisfies these requirements. This eliminates the possibility of mismatches in the task decomposition. Another risk could be that the expertise model does not satisfy the task input, output and goal. This risk can be reduced by formalising the expertise model using ML^2, [Balder et al., 1992] and checking its relation with leaf tasks.

1.5 Task model management

Building a task model requires a significant amount of effort. It is an investment that should be made in order to avoid certain risks in the KBS development process. The general rule is *not* to create a task model, unless there is a specific risk to avoid. We have formulated detailed guidelines in the following sections.

The states of the task model are described in an informal way. If the task model is described informally then it can only be evaluated empirically. If the task model has been described formally, then it would be possible to perform more automatic consistency checking. The following 'development states'[2] are considered relevant.

1.5.1 Development states

A task model consists of a structured set of frames, each representing a task-statement. All tasks have a number of instantiated attributes. The value of these attributes can can just be filled in, extensively described, or even validated.

A state of a single task can be represented by a triple: <Type of TM, relation/slot, relation/slot states >. The state of a task model is the collection of the states of the task statements it contains.

Development states of a task model can be defined in terms of these states; we propose the following development states: A task model is in a development state if all obligatory slots/relations (input, output, description, goal) of the tasks frames it contains are in the same state of development. For instance, a development state has been reached when all obligatory slots/relations of the task model describing the starting situation have been identified, or all obligatory slots/relations have been validated.

The following 'states of development' for the slots/relations have been identified (following [de Hoog et al., 1993]):

- *empty:* There is no information about the model component available.
- *identified:* the "names" of what the component is about is known.
- *identified:* a model component has been identified and given a unique "name".
- *validated:* there is strong evidence that the information in the attribute is correct with respect to a source. In the task model these sources include the actual organisation and the *Common*Kads models.

Important states of development are:

- The task model of the existing situation is 'sketched': < initial TM >, < name, description, input, output >, < identified >.
- The task model of the required situation is identified: < required TM >, < name, description, input, output >, < identified >.
- The task model of the required task model is described: < required TM >, < name, description, input, output >, < described >.
- The task model of the required task model is described and assigned : < required TM >, < name, description, input, output, agent >, < described >.

[2] Also called landmark states.

These development states define the state of information included within a task model. They do not refer to information omitted from a task model. Therefore, the 'correct level of decomposition' cannot be represented by the states above.

The question if the model has the correct level of, or the right type of decomposition depends on factors like the problem, expertise- and organisation-models. In order to provide necessary and sufficient conditions for these states we have to extend the state description with references to states of other models. Creating formal definitions of these conditions is a complex job and reduces but does not eliminate their subjectivity. Therefore we define the following modelling states by informal relations with states of other analysis models:

- **expertise** (identified, described, validated): The problem solving required by the task model can be realised. This means that an expertise model has been created for the leaf tasks assigned to the KBS or there are other reasons to believe that the leaf tasks can be realised (for instance, experience with other projects).
- **communication** (identified, described, validated) The communication of the ingredients to be transferred between leaf-tasks can be realised in practice.
- **organisation** (identified, described, validated). The tasks fit in with the organisation model.
- **agent** (identified, described, validated). The task distribution fits in with the agent model.
- **design** (identified, described, validated). The tasks can be realised with the design model.

A task model that seems to be correct can appear to be of insufficiently detailed later in the KBS development process. The states of the task model are therefore not absolute but relative to a situation in the project.

1.5.2 Task modelling guidelines

This is an initial and unstructured inventory of modelling guidelines based on the risk driven development approach. The format of the guidelines is taken from [Van de Velde et al., 1993].

1.5.2.1 Guidelines that motivate task model development

1. **Situation** The tasks for the system are unclear.

   Objective To build a system that performs tasks in the organisation that need improvement.

   Risk A system is constructed that does not improve the organisation.

   Target state $<$ existing TM $>$, $<$ name, description, input, output, agent $>$, $<$ described $>$. Task model of existing situation.

Rationale An analysis of the current situation in the organisation is required to identify opportunities for improvement.

2. **Situation** The tasks for the system and user in the new situation are unclear.

   Objective To have definitions for user's and system's in a new situation; to assess the task distribution.

   Risk A system is constructed that performs tasks that are not part of the problem.

   Target state < required TM >, < name, description, input, output, agent >, < described >, plus ***realisable***. The leaf tasks of this task model should be assigned to agents. Tasks assigned to the KBS should be realisable in Expertise and Design Model.

   Rationale Requirements on the task to be performed by the KBS are not completely specified by the client and are taken as part of the project.

3. **Situation** Project scope is being defined.

   Objective To have the tasks for the future users and the system set.

   Risk It is unclear who is going to do what in the required situation.

   Target state < required TM >, < name, description, input, output, agent >, < described >. Elaborate the task model of required situation an make the task distribution.

   Rationale This will clarify what the KBS can do and what the users will have to do.

4. **Situation** Organisation model requires information about tasks in organisation.

   Objective To complete the organisation model.

   Risk Organisational model will use incorrect task information or cannot be developed.

   Target state < type as requested >, < attributes requested >, < status as requested > Develop a task model in direction requested.

   Rationale The task model provides the interface between the organisational model and the actual activities in the organisation.

5. **Situation** The expertise model for the tasks assigned to the system is being developed.

   Objective To realise an expertise model for tasks assigned to the KBS.

   Risk No expertise model can be developed for tasks assigned to KBS.

   Target state Relation task - model expertise model validated.

   Rationale The development of a new task model can make the development of an expertise model possible.

6. **Situation** Design model is being developed.

   Objective To develop a design model for the task model of the required situation.

   Risk The design introduces new requirements or new task distributions, for instance to use a certain application specification.

Target state Model the new tasks, and check whether they are acceptable for the organisation.

Rationale Requirements and solution are not static in the development process, when the design model comes up with a good new task distribution, a modification of the task model can be considered.

7. **Situation** Communication model is being developed.

   Objective To have the communication of the ingredients of the tasks developed in the communication model.

   Risk Communication modeller cannot find acceptable ways to shape the communication.

   Target state A new task distribution that can be validated. Try to find new task distributions or decompositions that group different activities.

   Rationale Requirements and solution are not static in the development process, when the design model comes up with a good new task distribution, a modification of the task model can be considered.

1.5.2.2 Guidelines that motivate suspending task model development

This is also an initial inventory.

1. **Situation** Task model is being developed.

   Objective To develop a task structure of current situation. To have a task model that can be used for KBS development, the activities in the organisation should recur at some level of abstraction. Only then we can come up with a stable task structure.

   Risk No stable task structure can be identified in the organisation.

   Target state Suspend task analysis of the current situation. Try organisational analysis and/or re-organisation. Otherwise try to develop some functional prototypes that support simple small tasks and test them within the organisation.

   Rationale Some organisations have no clear task structure and it is not always useful to force a task model upon their activities. In these cases, the task model should not be further developed and alternatives should be tried. One possible solution is organisational analysis. Another possibility is to develop prototypes and to analyse how these are used in the organisation.

2. **Situation** Task model is being developed.

   Objective To make a task model of required situation.

   Risk Leaf tasks cannot be assessed.

   Target state Task model link to expertise model: *identified*. Develop the expertise model in more detail and return to task model once it is clear whether or not the task model can be further decomposed or assigned to an agent.

Rationale It can happen that it is unclear if decomposition should stop, when for instance, it is not clear that a KBS would be able to perform a task.

3. **Situation** The leaf tasks are all assigned in the task model of the 'required' situation, and all obligatory attributes are identified and relation with expertise, design, communication and agent has been validated.

   Objective To complete the task model.

   Risk No risks available that can be addressed with task model development.

   Target state Task model is ready.

1.5.2.3 Guidelines that motivate making a new version of a task model

This is also an initial inventory.

1. **Situation** Task model of the existing situation has been elaborated uptil a certain level of detail and it is unclear whether to continue or not.

   Objective To develop a task model of the existing situation.

   Risk Task model of existing situation is developed in too much detail.

   Target state Suspend development of the task model and of the current situation and start development of a task model of the required situation.

   Rationale It is not useful to make a too thorough analysis of the current situation. Task analysis consumes resources and should therefore stop as soon as possible. Once a task model of the required situation has been worked out, some more analysis on the current situation could be carried out, if necessary.

2. **Objective** To realise the required task model in other *Common*Kads models.

   Situation Tasks assigned to expertise, design or communication model cannot be realised.

   Risk The project will fail to realise the required task model.

   Target State Make a new version of the required task model with alternative decompositions or distributions.

   Rationale It is difficult to foresee all consequences at the moment the task model of the required situation is created. It may happen that the instantiation of other models shows that certain things in the required situation should be changed.

3. **Objective** The creation of a task model of the required situation.

   Risk There is not enough knowledge about the task structure to complete the task model of the required situation.

   Target state Stop developing task model of the required situation and elaborate the task model of the existing situation.

   Rationale Task modelling of the existing situation is stopped as soon as the analyst has the idea (s)he knows enough of the situation. Later it may appear that more detailed information is required.

1.5.3 Task features

In addition a task model can be used to make an assessment of a situation. The assessment of the task model can be used to construct a 'desired' task model by providing information to apply the following general task assessment criteria:

- *goal reached:* Is the task reaching the task goal?
- *errors:* Are there errors made in task execution?
- *uniformity:* Does every expert perform the task in a uniform way?
- *stability:* Is the task performed with stable results?
- *accuracy:* Is the task reaching its goals accurately?
- *expertise sharing:* Do may people have the expertise to perform the task?
- *flexibility:* Can a change in input, output or goal be coped with?
- *response time:* Is the task performed fast enough?
- *inefficiency:* Have unnecessary tasks been performed or have any tasks been performed in an inefficient way?
- *consistency:* Has information used by the task been kept consistent?

More task features are listed in [Aamodt et al., 1992]. Once the task structure has been assessed, the analyst can decide to construct a new task model to solve problems identified (and perhaps a new organisational and agent model). The process of designing this new task model is by no means straightforward! The task model allows this result to be represented, but the problem solving has to be done by the analyst.

Chapter 2

Case study

In this section we show the results of applying the task model as part of a real project. This analysis was part of a study performed in 1992 at the AI lab of the Free University of Brussels. It was carried out to discover opportunities for knowledge based systems technology in aviation crew scheduling.

The *Common*Kads methodology prescribes a spiral development process. Therefore we have to consider several other *Common*Kads models to explain how the task model can be used in the different cycles. This case study shows in detail the development of the task model and refers to the development of other models in abstract[1]. Table 2.1, at the end of this chapter, shows which models have been developed during this project.

We first provide a statement of the problem, then a first version of the task model like an initial task analysis, then a second version as a requirements definition and finally a third version which is again a requirements definition that takes discoveries of the design model into account.

The inconsistencies you will find in this case study are partly due to the difficulty of working with so much related but distributed information. It was, in particular, difficult not to loose track of the different versions of the models. This case study has been created without the support of a workbench, because at the time, the *Common*Kads workbench did not cover the communication model and covered only parts of the expertise model. A tool like the *Common*Kads workbench complete with document generation facilities, will make such an analysis much easier and the results more precise.

This case study has been the first to cover most of the models. More cases need to be analysed in order to refine the model definitions and modelling guidelines, and to collect the experience in order to guide novice *Common*Kads users.

[1]This chapter is an updated version of the case study in [Duursma et al., 1992]; readers interested in the development of the other models for this case study can consult that document, though it is not up-to-date with the latest model definitions

2.1 Statement of the problem

Every season an aviation company has to schedule the aviation crews to cover a flight program. This task is performed by the 'scheduling department'. The scheduling department consists of sub-departments; one for planning and one for elaboration. Two experts in the 'planning department' estimate the number of crews. They handle this task by first making a schedule to cover the required flights. This schedule is not for anyone in particular, but for so called 'logical crews' (an abstraction from the actual crews available). These logical crews should comply with aviation laws and company rules. In a second stage the 'elaboration department' creates a schedule with the actual crew available and tries to keep the number of crews required below the estimation of the planning department. Although sometimes there is a match between the logical and actual crews, the goal of planning is to find out the maximum number of crews required and elaboration defines the actual schedules.

The experts of the planning department stated that their workload is too high. They complained especially about having to re-plan due to continuing changes in aviation laws, required flights and required equipment. The aviation company consulted us to see whether we could provide support.

2.2 First cycle: task model for problem analysis

We started by identifying the following 'objectives' and 'risks';

1. The client had not provided us with requirement specifications.
2. There was no idea what crew scheduling entailed.
3. It was unclear where the problems in the organisation were.

This was the motivation to start the development of a task model and an organisational model of the existing situation. In this deliverable we focus on the task model.

The goal of the scheduling department can be labelled "crew scheduling"; they have to take care of the fact that all flights are performed taking all laws and regulations into account, as economically as possible and with the consent of the aviation personnel.

This task was decomposed (by the organisation) into 'logical planning' and 'elaboration', tasks carried out by the two departments. Logical planning entails finding the amount of aviation personnel required to cover the flight program. 'Elaboration' is the assignment of actual personnel to actual flights.

The rationale behind this first decomposition was that the company wanted to know the theoretically optimal minimal cost solution as a measure to which they can compare the actual performance with, and it also needed to have a plan for the actual available personnel. Fig. 2.1 shows this decomposition relation between tasks we identified in the domain.

task *Scheduling*

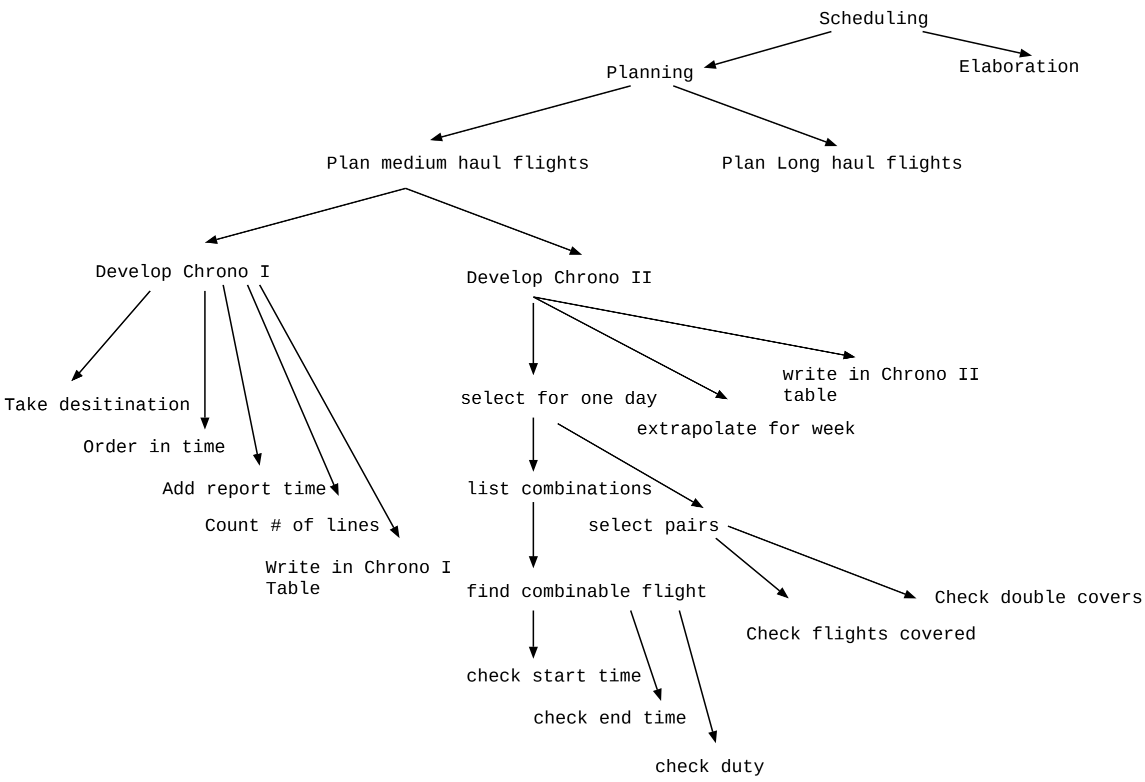

Figure 2.1: The sub-task relation of the task model representing the existing situation.

agent
Scheduling department.
input
Flight program.
output
Crew assignments.
goal
To cover all flights of the flight program with minimal costs. To assign actual personnel to the flights.
frequency
Every flight season (twice a year).
sub-tasks
Planning, Elaboration.
decomposition-type
Independent goals.

task *Planning*
agent
The planning department.
input
The flight program.
output

The amount and type of personnel required and a schedule for logical crews.

goal

The goal is to create a schedule with minimal cost that respects all aviation regulations. The amount and type of personnel should be able to perform in this schedule.

sub-tasks

Plan medium haul, plan long haul.

super-task

Scheduling.

task-decomposition-type

Independent goals.

The next decomposition identified is the separation of the planning task the planning of medium haul and long haul flights:

task *Plan medium haul flights*

agent

The planning department.

input

The medium haul part of the flight program.

output

The minimal costs and the optimal plan.

goal

Same goal as super-task.

sub-tasks

Plan one day medium haul, Plan two days medium haul, Plan week medium haul.

super-task

Planning.

decomposition type

The requirements for different days within the same week are usually the same, so when the task has been solved for one day, the result can be extrapolated for the whole week. This decomposition is based on extrapolation of a solution of a sub-task over similar sub-tasks. The two day plan is only for flights that have a two day cycle because they stay over at their flight destination and return next morning. (extrapolation type)

task *Plan long haul flights*

agent

The planning department.

input

The long haul part of the flight program.

output

The minimal costs and the optimal plan.

goal
Same goal as parent task.
sub-tasks
Not included in this analysis.
super-task
Planning.
decomposition type
Not included in this analysis.

task *Develop Chrono-I*
task type
Sorting and Summing task.
input
Flight schedule.
Order in departure time.
Number of flights.
output
Chrono-I. Chrono-I lists all flights per
day, ordered according to their departure time. It also
includes the report time of the crew.
Number of flights in the flight program.
goal
To produce a sorted structure that matches the application
of the sorting criteria on the input data.
Creation of the sum that is the sum of the input data
according to the summing criteria.
sub-tasks
Not decomposed because completely assigned to the system.
super-task
Plan medium haul flights.

task *Develop Chrono-II*
task-type
Combination + Selection
input
Chrono-I
Aviation laws
output
Chrono-II
goal
Chrono-II covers Chrono-I with minimal amount of
crew but within aviation laws.
sub-tasks
select for one day, extrapolate for week, write in Chrono **ii**
table.
super-task
Plan medium haul flights.

task *select for one day*
task-type
Selection
input
Chrono-I
output
Flight combinations for one day of the week.
goal
Identify optimal flight combinations for one day
of the week.
sub-tasks
list combinations, select pairs.
super-task
Develop Chrono-II-II
decomposition type
generate and select.

task *List combinations per flight*
task type
Pair combination.
input
The flights for a certain day in the week.
Start time and closing time of the flights.
Maximum duty.
output
set of possible combinations for a day.
goal
Creation of pairs that satisfy the pair constraints:
Combinations of flights that allow sufficient time but not
too much time in between and that are in accordance with the
flight regulations for aviation personnel.
sub-tasks
find combinable flights,check end-time, check duty.
super-task
find combinable flight.

task *Select pairs.*
task-type
Selection task.
input
A set of pairs of flight combinations.
Distribution of duty.
Minimum crew required to cover all flights of the season.
Experience about unsuitable combinations.
output
Selection: A list of flight combinations.
goal

To create a selection that satisfies the selection criteria.

sub-tasks

Not yet decomposed.

super-task

Select for one day.

task *Extrapolate for week*

task type: Extrapolation task.

input

The flight pairs for a certain day.

output

Extrapolation: A list of pairs for a whole week.

goal

The creation of the extrapolation of the instance over the extrapolation region.

super task

Develop Chrono-II

sub-tasks

Not yet decomposed.

We also decided to perform an analysis of the time (time-line analysis) spent in performing different activities. Fig. 2.2 shows an example of the analysis for task 'Make Chrono-I' per flight where about 40 to 120 flights are required.

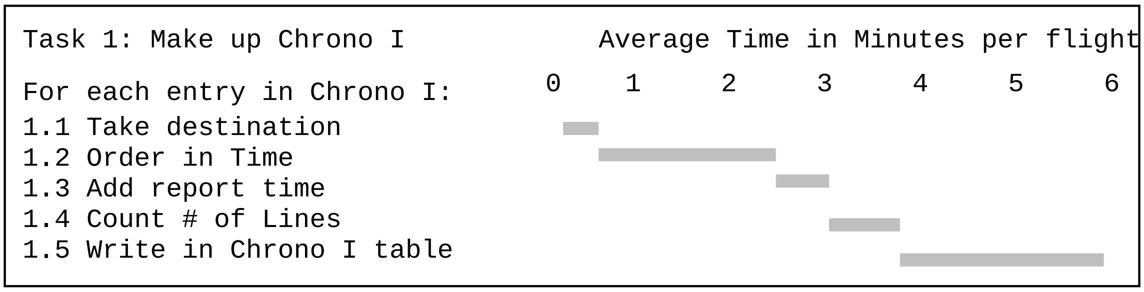

Figure 2.2: Time line analysis of task 'Make Chrono-I'

At this point in the task analysis process it was hard to decide whether to continue the decomposition or not. Therefore task model decomposition of the 'existing situation' was suspended. A risk driven modelling guideline motivates the development of a task model of the required situation.

2.3 Second cycle: Task model of required situation

In this section we enumerate the requirements to the system to be built. The required task decomposition changes the existing task structure somewhat due to different capacities of the introduced agent (i.e., the system). The goal of changing the task structure is to take advantage of the capacities of the different agents. From the time-line analysis and from comments of the experts we learned that the filling and modification of Chrono-I was one of the most time consuming tasks. This could be improved by a good screen editor, so it was useful to describe this in detail in a communication model. The development of the 'required' task model needed to be carried out in parallel with the required communication model, because the decomposition of the tasks depends on the assignment of the tasks.

We kept the existing first-level-of decomposition of the scheduling task in planning and elaboration tasks. The next level decomposition into medium and short haul flights of both the planning and elaboration task could also be kept. Changing these decompositions would have involved making changes in the department structure and this was outside the scope of our project.

We decided to assign the development of Chrono-I (a task that involved lots of writing and sorting) to the system. This task should then be elaborated to a design, either by conventional means or within the expertise model. Also task 'list combinations per flight' was assigned to the system. We were not sure whether it would be cost-beneficial to assign task 'select pairs' completely to the system. Therefore it was been assigned to both system and user but it required further elaboration of the expertise model. The tasks 'extrapolate for week' and 'writing up of Chrono-II' were assigned to the system. A task-decomposition was foreseen as shown in figure 2.3. The task distribution is fixed.

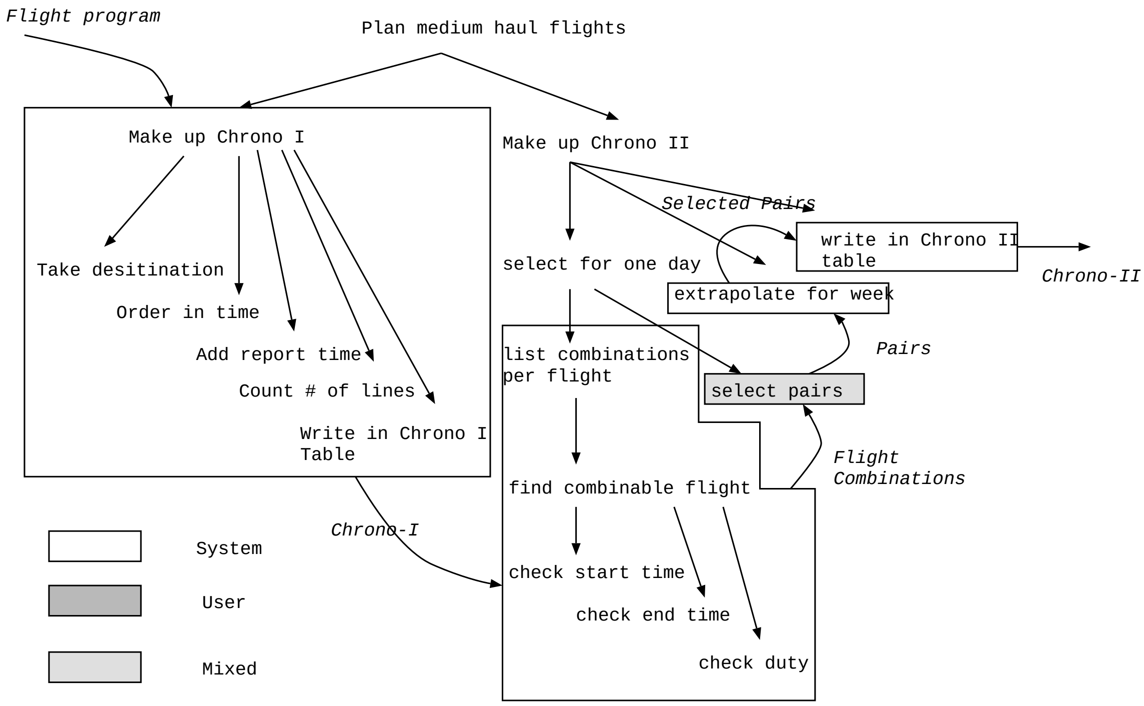

Figure 2.3: Required task decomposition and distribution for the planning of medium haul flights.

2.4 Third cycle: Improved task model of required situation

The task 'Select pairs' was elaborated in the expertise model as a step towards its design. We were convinced that the expertise model satisfied the goal of the task. Therefore we decided not try to prove that the algorithm was formally correct w.r.t the goal specification of the task. There was however a *risk* that the task-goal was incorrectly specified, i.e. that the task-goal did not map completely on the activities actually carried out by the experts. In order to be more certain, we decided to implement the algorithm we applied and compare it to the solutions made for some samples by the experts. We implemented the algorithm first in a prototype fashion. It then appeared that the algorithm did not produce exactly the same flight combinations as the expert had selected. This difference was not due to the random element in the algorithm, but some other factors, like unsuitable combinations due to frequent flight congestion or weather conditions. We then had the following choice:

1. To improve the goal definition of the task
2. To leave the goal definition as it is, but create a new task to refine the result.

We selected the second option, and decided to make a further decomposition of this task into a 'propose flight pairs' and a 'revise flight pairs step. The propose task could be assigned to the system, because the constraints to it are explicitly known (the communication model needed to be updated at this point). The revise step will be assigned to the expert user, because the criteria for the revision are unclear.

The task 'select pairs' is split up into a task 'propose flight pairs' and a task 'revise flight pairs', both new tasks in the environment. This is an addition to the required task-model (results in task model version 3).

task *propose flight pairs*
agent
System
input
Input set: A set of pairs of flight pairs.
output
Proposal: Proposed flight pairs.
goal
The Proposal should create pairs of a first and a second element, where each pair contains at least one element, all pairs together cover all the elements in the input, and there is no output possible with more double pairs (i.e. output is an optimal combination).
super-task
select pairs

The following task is created for the user:

task *revise flight pairs*

agent
user
input
Proposal: Selected flight pairs.
output
Revised pairs: Selected pairs that have been revised by the expert.
goal
To create a flight pair combinations that take all
experience with good combinations into account.
super-task
select pairs

Information in the agent model should be updated to reflect the responsibility of the (expert) user for this task. The communication model should also be updated in order to represent the information about the initiatives etc of these two tasks.

2.5 Summary

In this chapter we have shown how task models can be applied in a spiral-development of a system. We have produced a task analysis, a requirements version of the task model and an updated requirements version for a real project.

We have shown that the development of the models is not linear, but that discoveries in the development of one model or an implementation may change the development of other models.

Task	Communication	Expertise	Design	Stage
v1		v1		Analysis
v2	v1			Requirement identification
v3	v2	v2	v1	HL Design

Table 2.1: Model versions developed in this case study

Table 2.1 summarises what versions of the different models were developed during the project, although the task models are discussed in this document. We started with a task analysis (task model v1) and a very shallow expertise model (domain layer only, v1). Then we developed a required task decomposition (task model v2) and a communication model (v1); this can be interpreted as a software engineering requirements definition. Finally we made a high level design and developed a new version of the expertise model (v2) and a prototype. This made us aware of the need to modify task (v3) and communication (v2) model. And we also developed a first version of a design model.

Bibliography

[Aamodt et al., 1992] Aamodt, A., Benus, B., Duursma, C., Tomlinson, C., Schrooten, R., , Top, J., and Van de Velde, W. (1992). Task features and their use in *Common*Kads. Esprit project p5248, VUB AI lab, Pleinlaan 2, 1050 Brussels, Belgium.

[Balder et al., 1992] Balder, J., Akkermans, H., Aben, M., van Harmelen, F., Schreiber, G., and Wielinga, B. (1992). Formal methods for knowledge modelling in the CommonKADS methodology; a compilation. Technical Report KADS-II/T1.2/TR/ECN/014/1.0, ECN.

[de Greef and Breuker, 1990] de Greef, P. and Breuker, J. (1990). A methodology for analyzing modalities of system/user cooperation for KBS. In *Proceedings of EKAW-89: Third European Workshop on Knowledge Acquisition for Knowledge based systems.*

[de Hoog et al., 1993] de Hoog, R., Benus, B., Metselaar, C., and Vogler, M. (1993). The common KADS organisation model.

[Duursma et al., 1992] Duursma, C., Van de Velde, W., Schreiber, G., and Terpstra, P. (1992). Operationalising knowledge models. Esprit project p5248, VUB AI lab, Pleinlaan 2 Brussels, Belgium.

[Mcgraw and Herbison-Briggs, 1989] Mcgraw, K. L. and Herbison-Briggs, K. (1989). *Knowledge acquisition, Principles and Guidelines.* International Editions. Prentice Hall.

[Rademakers, 1991] Rademakers, P. (1991). Task analysis of an equipment assignment problem. AI-MEMO 91-1, V.U.B AI laboratory, Pleinlaan 2, B-1050 Brussels, Belgium.

[Shannon, 1980] Shannon, R. (1980). The validity of task analytic information to human perfromance research in unusual environments. In *Proceedings of the 24th Annual Meeting of the Human factors society.*, Los Angeles, CA.

[Sommerville, 1989] Sommerville, I. (1989). *Software Engineering (Third Edition).* Addison Wesley, Bonn.

[Steels, 1990] Steels, L. (1990). Components of expertise. *AI Magazine.* Also as: AI Memo 88-16, AI Lab, Free University of Brussels.

[Van de Velde et al., 1993] Van de Velde, W., Duursma, C., and Schreiber., G. (1993). Design model and process. KADS-II KADS-II/M7/VUB/RR/064/1.1, V.U.B AI laboratory, Pleinlaan 2, B-1050 Brussels, Belgium. V1.1.

[Woodson, 1981] Woodson, W. (1981). *Human factors design handbook.* McGraw-Hill, New York.