The SEC-System
Reuse Support for Scheduling System Development

Wout van Wezel
René J. Jorna

Faculty of Management & Organization
University of Groningen
P.O. Box 800
9700 AV Groningen
The Netherlands

E-mail: w.m.c.van.wezel@bdk.rug.nl, r.j.j.m.jorna@bdk.rug.nl
Phone: +31 50 3637181
Fax: +31 50 3632032

SOM theme A: Multi-level Interactions Within Firms: Primary Processes

Abstract

This article states that the development process of computer systems for task support of planning and scheduling is no unique activity that must be performed from scratch each time. Reuse of previously created conceptual and technical products is possible. We claim that adequate computer support starts with a task analysis. This analysis reveals general as well as unique aspects. The general aspects can in principle be reused. However, a taxonomy is necessary that separates general from unique aspects in the problem domain and task performance. We propose the Scheduling Expertise Concept (SEC) as such a concept for planning and scheduling tasks. This taxonomy is implemented in a computer system, the SEC-

system. The SEC-model and SEC-system enable faster and more accurate development processes in the planning and scheduling domain.

1. Introduction

Reuse of conceptual and technical information that is obtained during the development process of software gains increased attention (van Genuchten, Bemelmans & Heemstra, 1993). Generally speaking, reuse can be applied to everything that is, to some extent, structured. When this principle is used in practice, however, all kinds of restrictions arise. Programming languages must be the same or at least comparable, functional and technical specifications must use a common framework and users must execute comparable tasks. Despite these constraints, reuse in the development process is attempted to an increasing degree. This article discusses how we think to apply reuse to the development of decision support systems for planning and scheduling.

Section 2 deals with several reuse support approaches that are described in the literature. This will give some requirements and guidelines that we apply in our approach. Section 3 discusses the domain of planning and scheduling. There we advocate that scheduling support in practice needs an interdisciplinary approach. Section 4 describes several approaches that somehow are engaged with scheduling systems development and/or reuse. There we conclude that none of the discussed approaches complies completely with the requirement that are discussed in Section 2 and Section 3. The reuse requirements and scheduling support requirements are combined in Section 5. The resulting SEC-model is partially implemented in the SEC-system, that is introduced in section 6. In Section 7 we draw some conclusions.

The general idea behind the SEC-model and SEC-system is the following. A scheduling support system is based on an extensive analysis of the task context and task performance of human planners. Such an analysis reveals general and unique aspects. The general aspects are eligible for reuse. For this, however, we need a taxonomy, the SEC-model, that structures or categorises the general aspects in the analysis of the domain, the task execution, and the development of the resulting system. In this way analyses and designs can be stored and retrieved. This speeds up the analyses of new situations and the development of new scheduling support systems.

2.Reuse support approaches

Reuse is an important topic in software development. In reusing existing products in the development process, costs are decreased and accuracy is increased. Reuse can be applied to domain knowledge, design efforts, architectures, requirements, code, and documentation (Biggerstaf & Richter, 1987). Reuse also enables prototyping, since developers can get faster and more frequently feedback from users. Numerous approaches that formalise and implement reuse principles are described in literature. We will describe the approaches that are used in the SEC-model.

Reuse can be realised in two different ways: (a) generation and (b) composition (Biggerstaff & Perlis, 1989). The demarcation lines, however, are not very rigid.

Generation based techniques take three forms: (a1) language-based systems, (a2) transformational-based systems and (a3) application generators.

(a1). Language-based systems use an abstract language, consisting of a syntax, basic elements, a semantics and concatenation rules. Two requirements have to be met. The first is that one has to deal with a domain that with respect to its content is exhaustively assessed. The second relates to the language itself. In case slightly unknown facts turn up, the language should be easily extendible, just as is the case with the natural languages. Several developing groups are working on this perspective (Fox et al., 1996: the PROMPT-project; Breuker & Van der Velde, 1994: the KADS-project). Until now the theory seems to be very promising, but no effective and easily applicable tools have been realised, yet.

(a2). Transformational based systems focus on the transformation of models in the development process. This approach concentrates more on reuse of phases and operations to transform high-level specifications into programs, and less on the content of the models that are used.

(a3). The last category of generation based techniques consists of application generators. These systems embed knowledge of the target domain in their architectural design. With application generators, the domain must be described and implemented exhaustively beforehand.

An example of a generation based approach is the Draco-paradigm (Freeman, 1987). Draco is a general generation based reuse support approach that uses static domain languages, although these languages can be extended or split up to form a new language. A domain analysis must reveal the apparent objects, operations and relationships thereby formulating a theory of that domain. A so called *domain designer* specifies a language that covers the domain theory. This language is implemented in Draco. A *Draco applications specialist* uses this language to describe new systems. The *Draco systems designer* uses Draco and the descriptions to produce executable code. Draco is a generation approach, since the domain is determined and described beforehand, in contrast to composition based approaches that gradually and bottom-up form a domain by extending model bases with practical results.

In this article a composition based technique to the planning and scheduling domain is applied. Composition techniques merge existing and new components together to form a new application. New components can be added to the collection so that these possibly can be reused in subsequent development efforts. Most composition based reuse support approaches implicitly or explicitly use the concept of case based reasoning. Case based reasoning solves problems by adapting existing solutions that were used to solve comparable problems (Riesbeck & Schank, 1989). In general, composition based reuse support approaches specify how to store and retrieve cases. Those approaches differ in the primitives that are used to describe a case and the techniques that are used to find existing cases that are comparable to a new case. This is where confusion between generation and composition might come in, because the case based reasoning approach has strong roots in the case grammars from the 1970’s (Fillmore, 1968). In contradistinction to common grammars, case grammars have an explicit relation with the actions and goals of agents in real worlds. However, the description of grammar cases also requires some formalisation and depending on the definition of a language, the demarcation line between the generation and composition based approach may fade away. In the remainder of this section, we will describe some composition based reuse approaches of which ideas are applied in our approach.

Prieto-Diaz (1991) proposes faceted classification to process information that emanates from development processes. Model classification is done by choosing predefined keywords from multiple lists and linking these to the model. This method is different from singular classification where models are linked to one category and categories refer to each other. Singular classification results in a difficult to comprehend and difficult to maintain network of relations. Faceted classification needs three components: the classification system itself, a thesaurus to store the keywords, and a conceptual distance graph that contains similarities between keywords so that similar model components can be found. When searching, someone chooses keywords from the lists thereby aided by the thesaurus. After that, model components are selected that are the most similar according to the distance graph.

Ramesh & Raghav Rao (1994, p. 64) state that reuse support systems (RSS) “should be able to find and retrieve components according to given criteria, evaluate the retrieved components according to given constraints, and rank the evaluated components in some order”. They describe software components with the three tuple <action, object, medium>. Applicable components can be found by querying a database. As with the faceted classification approach (Prieto-Diaz, 1991) the query can be improved by providing a thesaurus and closeness factors. In addition, rule bases can search components and present them to the user.

Liang (1993) and Liang & Konsynski (1993) describe how analogical reasoning and case-based learning can be incorporated in model management systems. They state that analogical reasoning can be used to identify similarities between cases. Consider the analogy A is to B as C is to D. Analogical reasoning involves three operations. First, the *inference* operation derives the relationship between A and B. Second, A is *mapped* to C. Third, the information that is revealed by the inference and the mapping is *applied* to D. This reasoning process can be used to incorporate a case-based approach in a model management system. Liang states three key issues for this. First, cases, which are a combination of a problem (A and C) and a model (B and D), must be represented and stored. Second, it must be clear how to identify similarities between models and problems, and third, it must be evident how new models can be constructed from existing ones. Liang and Konsynski state that the modelling languages must incorporate problem features that allow fast recognition of similarities and differences. In addition, the inference and application operations require fluent transformation from problem to model. Liang (1993) uses graphs to operationalise these issues. In contrast to the aforementioned approaches, Liang does not explicitly relate models to software components. His approach is mainly oriented towards quantitative models. The general principles, however, do not exclude appliance of these ideas to software reuse.

Kwon & Park (1996) describe the Reverse Modelling Tool (RMT), a modelling support system in which reuse is based on reverse modelling. They claim that a model is an aggregation of model constructs. These constructs must be extracted and put in a model base. When a new problem is analysed, existing components can be composed to form a new model. A meta-system enables application of RMT in various domains. The meta-model determines the scope of the models under consideration in an object-oriented fashion. As Liang's and Liang & Konsynski's modelling by analogy, RMT is oriented towards mathematical models but again, the principles can be used for other problem domains as well.

Our reuse approach tries to combine several composition based methods. We adopt Liang's analogical reasoning mechanism and Kwon & Park's model decomposition principle. Furthermore, we use the faceted classification ideas of Prieto-Diaz and Ramesh & Raghav Rao by using a thesaurus and closeness factors to find analogical models. These ideas are used in a functional description and partial implementation of the SEC-system that supports reuse of scheduling systems. First, however, we will discuss some general aspects of the scheduling domain and a number of scheduling support approaches from literature in order to clarify the domain we are interested in.

3. The reuse domain: planning and scheduling

Planning and scheduling are subjects of interest in various scientific areas and hence it is difficult to find or formulate unequivocal demarcations of the problem domain (Jorna et al., 1996). We acknowledge the differences between planning and scheduling, but nonetheless we use both terms to denote the same class of problems, i.e. the problem that human planners in an organisational context face. Although planning problems are considered to be at an abstract level, whereas scheduling problems are at the operational level, we would like to argue that, from the point of view of a human user, planning and scheduling can be approached in the same manner. This section reviews several scientific areas that engage in planning and scheduling. This review will make clear that a method for scheduling or planning support development needs an integrated approach that incorporates several aspects of planning. The SEC-model, that will be discussed in the Section 5, combines the requirements for scheduling support that are imposed by the various research fields.

From a psychological point of view, planning is a mental activity besides problem solving and perceptual and locomotive skills (Card, Moran & Newell, 1983). People establish an order or arrangement on how to solve problems beforehand. The degree of planning depends on the complexity of the problem and the kind of task. Mental arithmetic hardly requires planning, whereas for example a game of chess requires much planning. In this view, planning is part of a larger task. Planning, however, can also be a task itself. In this perspective, planning is a synthetic task since the outcome is a design (Clancey, 1985). Important sub-tasks of planning are for example clerical work, counting, negotiating and problem solving (Mietus, 1994). Clearly, planning as a task (like other tasks such as chess or mental arithmetic) incorporates planning as a mental activity. The most apparent distinctive determinant between the planning task and the mental activity is the question of who or what will execute the plan. Somebody who mentally creates a plan will use this plan himself to perform the larger task. Contrarily, the outcome of a planning task is often used by others in their own task context.

Planning in management theory deals with the allocation of resources, for example money to investment projects, products to machines, or staff members to shifts. Often, the planning in an organisation is performed by more than one planner. In these cases, the organisation of the planners is important, i.e., the division of tasks and responsibilities between planners. Management theories often divide planning in several layers in order to cope with the complexity of planning. Within those layers distinctions can be made in time horizons, time preciseness, scarcity of objects, geographical preciseness, et cetera. Anthony (1965), for example, distinguishes strategic planning, tactical control, and operational control. Strategic planning has a long time horizon (more than five years) with abstract decisions, tactical control deals with medium term decisions (one to five years), and operational control is about short term concrete decisions.

Production management planning and control frameworks not only have separate planning outcomes for different time horizons but also for different entities such as machines, products, operators, and stock. Most other planning types, for example employee scheduling and route planning, have no theoretical foundations with regard to the organisation of the planning itself. This is the main reason that the planning task in production companies is supported more often than the planning task of, for example, staff scheduling.

Planning is also one of the subjects of interest in Operations Research (OR). OR uses a mathematical approach to solve well defined scheduling problems such as job shop scheduling or the Travelling Salesman Problem. Often, scheduling problems are too complex to solve with optimising methods, such as linear programming or dynamic programming. Heuristics can be applied to find sub-optimal solutions for such complex problems. OR uses abstraction of objects to make an algorithm more generally applicable. When a satisfactory algorithm is developed for a well-defined class of problems, other problem types can be translated to this class. For example, to create a staff schedule, employees can be seen as machines and shifts as products. Then a job shop scheduling algorithm can be used. This mechanism creates the opportunity to make generally applicable algorithms.

The last field of research that we discuss here is Artificial Intelligence (AI). Within AI two fields can clearly be distinguished: AI planning and AI scheduling. AI planning deals with intelligent agents that must create a plan of their future actions in an artificial world. Planning agents are often based on general problem solving mechanisms thereby adding specific features for planning with respect to representations of goals, states, actions, and solutions (Russell & Norvig, 1995). AI planning is roughly comparable with simulation of planning as a mental activity.

In contrast, the field of interest of AI scheduling deals with the task of planners. Several AI techniques are used to solve scheduling problems. Results are reported with rule bases, neural networks, genetic algorithms, and simulated annealing. The mainstream, however, is concerned with constraint based programming techniques such as constraint logic programming and constraint satisfaction programming. AI scheduling methodologies divide into two areas: constructive methods and repair methods. Constructive methods start with an empty schedule and add assignments until the schedule is complete. Repair based methods try to adjust an existing schedule in order to remove conflicts or to further optimise the solution (Zweben & Fox, 1994). AI scheduling techniques gain interest because they use generally applicable techniques that find satisfactory solutions.

Decision Support Systems for planning and scheduling must focus on planning problems in practice. The perspectives and results that origin from the various scientific approaches that we discussed can be used in co-operation with generic decision support approaches to design these systems in a structured way. An integrated approach is necessary to cover all aspects of planning. By considering planning as a mental activity the system developer gains insight into the problem space of human planners. The planner will only accept the solutions of a computer program if he understands the results, i.e., the user and the system should have coincident problem spaces (Prietula et al., 1994). Efforts to combine scheduling support systems with the users perception of the problem have already resulted in various mixed-initiative scheduling approaches (Prietula et al., 1994; Veloso, 1996; Smith & Lassila, 1994; Smith et al., 1996).

The planning task in practice is more than just problem solving. Other task components such as clerical work and counting must also be supported. In addition, planners must often collaborate to make a schedule, and many times they have to negotiate about (soft) constraints and goal functions. Furthermore, the planning department itself must be organised by assigning tasks and authority relations. Planning support must take into account all these aspects.

Planning support systems can use techniques from OR and AI to generate schedules. The degree of structuredness of the problem under consideration determines what kind of techniques can be applied. Knowledge based AI techniques fit with the problem space of planners but often have to sacrifice performance with respect to optimisation. Constrained based techniques and OR algorithms are efficient and powerful but show a gap with the way human planners think.

All approaches of planning and scheduling somehow deal with arranging scarce resources. Examples of resources that were mentioned until now are chess pieces, money, vehicles, machines, staff, etc. More precisely, scheduling always concerns multiple tokens of different object types. Production schedules, for example, arrange products, machines, and time intervals, whereas staff scheduling deals with staff members and shifts. This common feature of scheduling problems is used in the SEC-model as the main modelling principle. We use the following definition to denote scheduling problems: *planning or scheduling is the attuning of instances of different object types, thereby upholding constraints and goals*. This definition imposes restrictions on the models in the development process. The object-oriented approach, however, also creates possibilities for reuse. This will be explained in Section 5. First, however, we will outline some literature that functions as a background for our modelling approach.

4. Modelling approaches for scheduling support and reuse

Several authors propose scheduling and planning modelling approaches that in some way deal with scheduling support and/or reuse. We stress that these authors not always have reuse as their goal. In this section, however, we look at their research mainly from a reuse perspective and therefore possibly neglect aspects that these approaches have as their main objective.

Verbraeck (1991) describes two model types for scheduling problems: an *m*-dimensional space and a transition representation. In the dimensional representation different resource types are put along axis in a diagram. A number of elements must be placed in this diagram thereby forming a schedule. Formally, this representation is a three tuple <S, E, C> where S is the *m*-dimensional space, E is the set of elements that must be place into the schedule and C denotes the set of constraints. The Gantt-chart is an example of this representation. In a Gantt-chart, machines and times distend the problem space and operations are the elements that must be placed in the diagram.

The second representation of scheduling problems that Verbraeck describes is a transition diagram. In this representation, a scheduling decision is depicted as a transformation from one state to another in stead of a placement in a diagram. Formally, a transition model is the six tuple <V, B, Λ, G, W, C>. V is the set of variables that describe the state of the system, B is the initial state and Λ is the set of admissible transitions. G is the goal state, W is the set of cost functions of transitions and C is the set of constraints. A transition is described by a condition that determines when the transition may be applied and the new values of the variables that are changed during the transition. The dimensional representation form is interchangeable with transition diagrams, though in most cases either one of them is the most appropriate.

Although Verbraeck (1991) presents a domain representation with which various planning and scheduling problems can be described, he does not describe a computer system that incorporates this language.

Woerlee (1991) defines a production scheduling decision as “how to process a set of activities *A* given a set of resources *R* on a time axis *t*”. Such a decision problem is characterised by entities, that in turn are characterised by attributes. Production scheduling entities are activities, resources, and time. A scheduling problem is modelled by a table in which the rows corresponds to elements such as activities and resources, and in which the columns contain the attributes. Constraints are defined on elements or groups of elements. These tables are used in a generic scheduling system. The system incorporates the structure of production scheduling models, but not the models themselves. This offers flexibility since the system can be used for different scheduling problems by mere adjustment of the tables. The shell that Woerlee describes includes a query language for the tables, a communication method for the analyst, and a visual interactive language for the user.

Van Putten et al. (1993) describe a generic user interface constructor for planning and scheduling applications. They propose four predefined graphical representational forms: a tabular view, a map, a Gantt chart and a bar chart. An Application Constructor creates a prototype application based on a datamodel with scheduling objects and their proposed graphical representation form. In their constructor, Van Putten et al. separate the model from the way the model is represented to bring about reuse of the user interface. A panel is used as an abstract construct that links a specific model to a representational form.

Wennink (1995) and Wennink & Savelsbergh (1996) describe a specification method for planning problems that is intended for use with a planning board generator (PBG). They define planning as the assignment of processes to resources and time intervals in such a way that all constraints are satisfied. The specification method uses nodes to represent processes and resources. The properties of processes and resources are modelled by attributes. Edges, arcs, and auxiliary nodes describe relations between objects. Wennink enumerates a number of basic properties of time, resources, processes and assignments to give content to his specification method. Properties of resource nodes are for example the name, availability periods, modes, usage, and consumption.

The approach of Wennink specifies a fixed set of elements and attributes beforehand. The analyst must translate real world objects into process nodes, resource nodes, and capability nodes and model the relationships in arcs. Wennink assumes that the attributes he describes are sufficient to attain adequate scheduling support although extension with new attributes is not prohibited. A so-called Planning Board Generator uses the description of nodes, arcs, and attributes to create a planning board with a user-interface that has representation and manipulation features, a constraint checker, and a schedule generator. The user-interface and constraint checker are created with the generation approach, whereas the schedule generator is created with composition based techniques due to the poor efficiency of generic problem solving algorithms.

According to Wolf (1994), scheduling problems can be characterised by the tuple (P, O, D, pr, op, S, f, g, t_{max}). *P* is the set of processors, *O* is the set of operations, *D* is the set of decisions, *pr* is the function that assigns the processor, *op* is the function that assigns the operation, S is the set of possible states, the function *f* describes state transitions, the function *g* describes the duration that a decision will be active and t_{max} defines the scheduling interval. In contrast to Verbraeck (1991) and Woerlee (1991), the framework of Wolf models constraints and quality measures separately from the tuple that describes the scheduling problem, because he assumes constraints do not possess the required homogeneity. Wolf also describes some primitive functions for the user interface. These functions operate on scheduling objects such as processors, operations, decisions and constraints. Wolf describes two levels of objects in the Decision Support System for scheduling. The first level consists of the super-classes of scheduling objects such as resource, decision, and constraint. This level contains the properties and methods that are common to all scheduling problems. The second level describes domain-specific information. The structure of a specific DSS is created by linking generic modules and domain-specific modules. The user of the system then creates object instances for the relevant resources, operations, and constraints.

Liu (1993) describes a Problem Formulation Framework (PFF) that guides knowledge acquisition in the scheduling domain. The framework has a number of domain independent slots that occur in all scheduling problems, e.g., factory scheduling, transportation scheduling, and manpower scheduling. Liu states that a scheduling problem is composed of a set of tasks, a world where the tasks are performed, and a set of objective constraints. A task is a plan that can be described with a set of primitive actions linked by a set of relation constraints, i.e., temporal, conditional, or disjunctive relations. A primitive action is described by the time and resources that are required, the effects of the action, and the constraints on time, resources, and effects.

The primitive actions are performed in the scheduling world. This world is composed of a set of resources, physical environment information, and a set of constraints on the environment. Resources are described by the initial capacities, a set of static properties, and a set of dynamic properties. Hence, Liu defines scheduling as “a process of assigning times to primitive actions in a set of plans and of reserving resources that actions need so that a set of constraints is best satisfied.” (Liu, 1993, p. 259-260). The PFF does not include task modelling, application modelling, or references to implementation in a reuse support system.

Smith et al. (1996) and Smith & Becker (1997) propose OZONE, a planning and scheduling toolkit that can generate constraint-based scheduling systems. This toolkit is based on a planning and scheduling ontology that cosists of demands, resources, operations, and products. The OZONE toolkit consists of object-oriented components that are based on this ontology. Based on features of the problem domain, OZONE provides an application skeleton that can be extended with problem specific requirements.

Sundin (1994) describes problem solving methods for assignment and scheduling problems for the CommonKads library. Like Schreiber (1993), he treats scheduling as a special case of the assignment task. Assignment is a synthetic task (Clancey,

1985) that operates on at least two sets of objects. The solution of an assignment task is a set of relations between objects of the different sets in such a way that the constraints are not violated. The roles of the objects differ in different domains. According to Sundin, the objects of scheduling problems are respectively activities and time-slots. In order to depict the more generic class of assignment problems, however, Sundin models one object set as resources and the other object set as components. Scheduling problems with more than two object sets, e.g., the classroom assignment problem with classrooms, teachers, classes, and topics, are decomposed into several assignment problems. Sundin uses the generic design problem solving methods *propose* and *revise* to solve assignment problems.

Henke (1994) describes an object-oriented approach for scheduling space shuttle missions. She describes a hierarchy of objects that are specific to space shuttle missions, for example the manifest, flows, activities, vehicles, and facilities. The Automated Manifest Planner (AMP) uses this representation to provide a planning tool with a flexible user interface that can automatically create schedules.

The application framework for automated planning and scheduling (ASPEN) of Fukunaga et al. (1997) is developed in the spacecraft control and operations domain. Their approach, however, pretends to be more generic. ASPEN is an object-oriented system with reusable software components that are commonly found in scheduling systems, such as a constraint modelling language, a constraint management system, a temporal reasoning system, and a graphical interface. The base components, together with a modelling language, cover a range of scheduling problems. For more specific problems, the code of the framework can be extended.

Ehlers & Van Rensburg (1996) describe a general eight layer scheduling model (GES). The layers are defined on the presence of the following parameters: (a) the number of products (either one or more), (b) the number of resources (one or more), (c) the number of processes (one or more), (d) the manufacturing timeframe (fixed period versus variable period), and (e) the constraints (none, due dates, due dates with priorities, or user defined). Ehlers and Van Rensburg state that each production scheduling problem can be fit into one of these eight layers. They classify scheduling problems with the following criteria that are based on the eight layers: sites, products, machines, processes, and manufacturing time period. Object-oriented programming techniques are used to implement these ideas in a generic applicable production scheduling system. This system is composed of a priority assignment class, a constraint representation class, a resource class, an event class, a configuration scheduler, a concurrent scheduler, and a system timer.

Tate (1993) and Beck & Tate (1996) describe a common framework for open planning and scheduling systems. This framework is based on their O-Plan and TOSCA systems, which are generic planning and scheduling tools that are composed of clearly distinguishable components. These components are mainly oriented towards constraint processing. Both O-Plan and TOSCA have the following components: domain information, plan/schedule states, knowledge sources, controller, and support modules. Decomposition and modularity of system components should enable reuse by distinguishing the architecture level from the plan or schedule representation.

Pillutla & Nag (1996) provide an object-oriented framework in which "real-world" objects of manufacturing scheduling problems such as machines, products, and raw materials are modelled with part-whole and generalisation-specialisation relationships. Information about objects types is stored in class templates, and the set of class templates constitutes the model dictionary. A model for a new planning problem is either derived from existing models in the model base, or created from scratch. In both cases, the new models can be added to the model-base to be reused in the future. Pillutla & Nag restrict themselves to the conceptual modelling of manufacturing problems but they acknowledge the need to enlarge the scope to both other problem domains as well as the design and development of the resulting scheduling support systems.

Although the authors that we discussed all somehow engage in planning support and reuse, the approaches differ significantly in scope and richness. The modelling techniques are mainly oriented towards description of the scheduling problem. The approaches include state space models (Verbraeck, 1991; Wolf 1994; Liu 1993; and Beck & Tate 1996), Kads domain models (Sundin, 1994), tables (Woerlee, 1991), activities (Fukunaga et al., 1997), complexity layers (Ehlers & Van Rensburg, 1996), and objects (Henke, 1994; Pillutla & Nag, 1996). Although Wolf, Ehlers &

Van Rensburg, and Smith et al. use object-oriented techniques to implement scheduling systems, they do not use object-oriented analysis to model the scheduling problem itself (Table 1).

Some of the articles that are discussed present domain specific models, i.e., they do not offer an integrated approach that explicitly incorporates modelling techniques for scheduling problems in different domains. Wennink, Woerlee, Wolf, Ehlers & Van Rensburg, and Pillutla & Nag limit themselves to manufacturing scheduling problems, and Henke and Funaka et al. only discuss applications for space craft missions. Due to the limited domain, these approaches can adopt the generation approach in their reuse efforts. Correspondingly, the modelling languages they use can only be applied to their successive domains.

The other authors describe more generic approaches. Verbraeck, Van Putten, Liu, Sundin, Beck & Tate and Smith et al. explicitly mention the domain independence of their approaches. However, to our opinion their articles lack semantic information about differences and similarities in different scheduling domains. That is to say the models of these approaches can not be both abstracted and decomposed. Hence, their descriptions are too generic and their techniques do not possess the functionality that is required by the reuse paradigms.

The majority of the approaches describe a modelling language and a scheduling system that can be configured by specifying a problem instance with the modelling language. This amounts to the generation based reuse approach. Some approaches indicate that the system can or should be extended with problem specific characteristics whereas other approaches assume that the modelling language can cover all problems in the domain (Table 1). Only Pillutla & Nag, however, indicate how the existing models can be extended by newly acquired knowledge about the domain.

None of the authors provides techniques to model the task of human schedulers. Sundin describes generic problem solving components that can reflect the task performance of humans but the Kads-approach is not focused towards lenient adaptation of the task models. Especially decision support in a complex domain like scheduling needs adequate task support. The SEC-model, that is discussed in the next section, removes the deficiencies with regard to reuse requirements and task modelling.

	Problem representation	Domain	Reuse approach	System aspects
Beck & Tate	State space	Domain independent	Decomposition and modularity of components	Standard components aimed at constraint processing; constraint specification is problem instance
Ehlers & Van Rensburg	Complexity layers	Manufacturing	Classify scheduling problem in one of eight layers	Generic object-oriented components
Fukunaga et al.	Activities	Spacecraft operations	Modelling language to define scheduling problems	System can be configured to a specific problem with the modelling language
Henke	Objects	Space Shuttle Missions	Object hierarchy	Object-oriented components and rule based approach
Liu	State space	Domain independent	Framework with predefined slots	None
Pillutla & Nag	Objects	Manufacturing	Object hierarchy	None
Smith et al.	Scheduling and planning ontology	Domain independent	Object-oriented framework based on a common ontology	Generic modules are implemented and can be specialised with domain specific features when needed
Sundin	Kads domain model with resources and components	Domain independent	Abstract concepts and generic problem solving methods	None
Van Putten et al.	Axis	Domain independent	Predefined graphical representation and separation of model from representation	System that creates an interface based on the defined axis and representation form
Verbraeck	State space	Domain independent	Scheduling specific modelling primitives	No generic system is presented
Wennink	Graphs/axis	Manufacturing	Scheduling specific modelling primitives	A system that can create a scheduling system based on a model is proposed but not implemented
Woerlee	Tables	Manufacturing	Scheduling specific modelling primitives	Generic system that can be configured with a production system model

Wolf	State space	Manufacturing	Separate description of generic scheduling objects and domain specific objects	Generic modules are implemented and systems can be made by implementing additional domain specific modules

Table 1. Modelling approaches for scheduling support and reuse

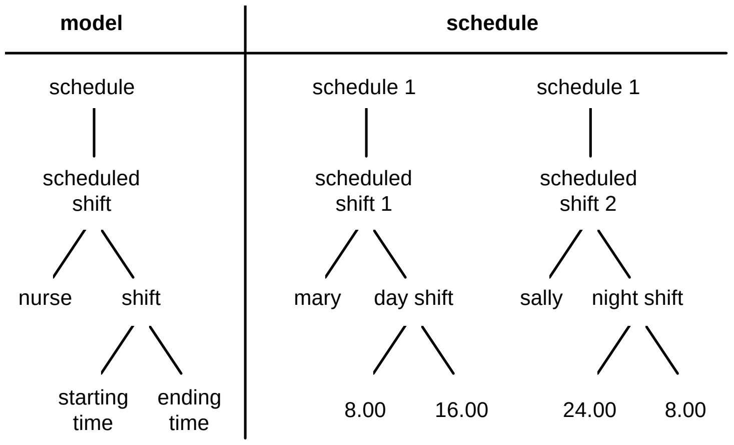
5.The SEC-model

This section describes the various models of the Scheduling Expertise Concept. These models are created in accordance with the reuse requirements that are described in Section 2. In addition, the models cover aspects that come from the various scientific approaches that were discussed in Section 3.

In our approach, we distinguish *domain models*, *task models*, and *application models* (van Wezel et al., 1996). The scheduling domain is represented with three model types. First, the arrangement problem is modelled with *object aggregation*. Several layers represent the aggregation levels as perceived by the schedulers. An object in our model is a construct from the schedulers perspective, i.e., it is a semiotic construct since it refers to signs as opposed to real world objects in an ontological sense (Jorna & Van Wezel, 1996). This releases us from the obligation to model the so called “real” world. From the point of view of the schedulers, it is their reality that we have to account for. This means that one has to work towards some kind of a to be agreed upon artefact – a model, domain or world – that is represented by signs and sign structures. The time one spends in realising such a consensus may be regained in minimising the after-effects of mis-representation.

Second, illegal combinations of object instances are depicted by *constraints*, and third, *goal functions* depict preferred combinations. An example of a nurse schedule is depicted in Figure 1. Objects are either *singular* (nurse, starting time, and ending time) or composed of other objects, i.e., *aggregate* objects (shift, scheduled shift, and schedule). The nurse schedule example consists of a number of scheduled shifts, and a scheduled shift links a particular shift to a nurse. Shifts themselves are composed of a starting time and an ending time. Assignments in the schedule are depicted by instances of objects.

Figure 1 also contains an example of a constraint. It states that student nurses may not work in the night shift. Therefore, if Sally in Figure 1 is a student nurse, the schedule is illegal. The goal in Figure 1 represents a preferred minimisation of the deviation of worked hours and the number of hours in the contract of the nurse. Of course, a domain model can contain several constraints and goals.



constraint: ∀nurses: ¬scheduled shift.nurse.qualification = student ∨ ¬scheduled shift.shift=night shift
goal: minimize abs(schedule.nurse.hours worked-schedule.nurse.hours in contract)

Figure 1. Example of object model

Generalisation of object models is performed in two ways. First, singular objects are abstracted by inheritance. Properties that are common to two object types are described at a higher ancestor object type. For example, the object type employee is an ancestor of both the object type nurse and the object type mechanic. These three object types show the differences and similarities between nurses and mechanics. The inheritance mechanism should not be restricted to singular inheritance. The object tree must be dynamically created when searching for similar objects. If, for example, we need for some reason the distinction between men and women in the tree and not between nurses and mechanics, single inheritance would result in four object types (male nurses, female nurses, male mechanics, female mechanics) in stead of the desired two (men and women).

At the highest level of abstraction we distinguish seven primitive singular object types: person, time, task, location, product, machine and vehicle. All occurring objects in the scheduling domain are presumed to have one of these object types as the ultimate ancestor. This classification is derived from a survey of scheduling situations in more than 50 companies (Bakker, 1995). We emphasise that this classification is not definite; other classifications are possible, since the classification scheme should remain in accordance with new scheduling and planning “realities”.

The second way of generalisation of object models is clustering of singular objects. In practice, only a limited set of models will occur (van Wezel, 1994). Staff scheduling, for example, often deals with persons, tasks, and time or persons, locations and time whereas production scheduling deals with machines, products and time, alternatively supplemented with, for example, persons or tasks. This taxonomy is a combination of primitive singular objects and aggregate objects. The availability of a set of object models that cover most situations can quicken the development process. We repeat that object models always must represent the schedulers perspective and not the analysts perspective. Especially aggregation objects can refer to constructs that do not exist in the reality of the analyst and therefore are difficult to grasp by outsiders.

The scheduling task execution is represented with two model types. First, *task decomposition* denotes the sub-tasks that comprise the scheduling task. To meet the transformation requirements, sub-tasks are described as operations on objects in the domain model. The highest level of task decomposition shows the sub-tasks clerical work, negotiation, attuning and adaptation. The attuning task deals with the actual assignment of object instances. This sub-task can be decomposed further in sub-tasks such as selecting, ranking, assigning and manipulation of constraints and goals. Second, the *task strategy* depicts the order in which the sub-tasks are performed (Mietus, 1994). The order of execution of the scheduling sub-tasks is, for example, administrate, attune, negotiate, adapt. The order within the attune sub-task could be rank, select, assign, manipulate. Of course, the strategy does not have to be solely linear. Iterative sequences of sub-tasks are more likely to occur, as was found by Mietus (1994).

By generalisation of task models, specific tasks are abstracted to task models that can be applied to multiple domain models. We follow three perspectives. First, due to their reference to objects, sub-tasks can be abstracted by referring to abstract object types. If, for example, a task model of nurse scheduling uses a subset of the available attributes that corresponds to the abstract object type “employee”, the task model can be used for, e.g., mechanics scheduling. A second way of generalisation of task models uses roles of objects. For example, a task can be described as matching supply and demand. This, however, requires other taxonomies for the domain than the object structure we propose. The third generalisation mechanism typifies scheduling tasks with respect to the task strategy. The strategy can be described with features such as periodically or continuously, batch or one by one, top-down or bottom-up et cetera (Van Wezel & Van Donk, 1996; Jorna et al., 1996). The combination of such features and predefined abstract task models will aid in analyses of scheduling tasks.

After composing domain and task execution models, the application components can be modelled. This means a shift in the development process from analysis to design. The Scheduling Expertise Concept includes a generic application taxonomy with components that can be used in any scheduling system. First, the *user-interface* presents the information to the user, e.g., an electronic planning board. A user-interface is composed of a number of components (widgets) such as menus and push-buttons (Eberts, 1994). The components are either linked to objects or sub-tasks. Push-buttons, for example, can be used to trigger the execution of sub-tasks. Often a matrix is used to depict the schedule, for example, the object type time on the x-axis, the object type nurse on the y-axis and the object type shift in the cells. If sub-tasks or objects are reused in the development process, possibly the accompanying user-interface components can also be reused. Second, the *inspector* examines what constraints are violated in a schedule. Third, the schedule is assessed by the *evaluator* with respect to the goals. Current research concentrates on the possibility of a generic inspector and evaluator by specifying a generic constraint language that is based on the SEC domain modelling technique. The fourth main application component is the *generator*. This component can, alone or in corporation with the scheduler, create schedules. The generator uses the decomposition of the scheduling task execution as it is performed by the human scheduler. In this way, planners are more inclined to accept the outcome of the system.

The application components must be implemented somehow. This means that choices for the development platform, the programming languages, etc. have to be made. Object-oriented languages suit the aforementioned object-oriented approach. The various models that constitute the SEC-model are depicted in Figure 2.

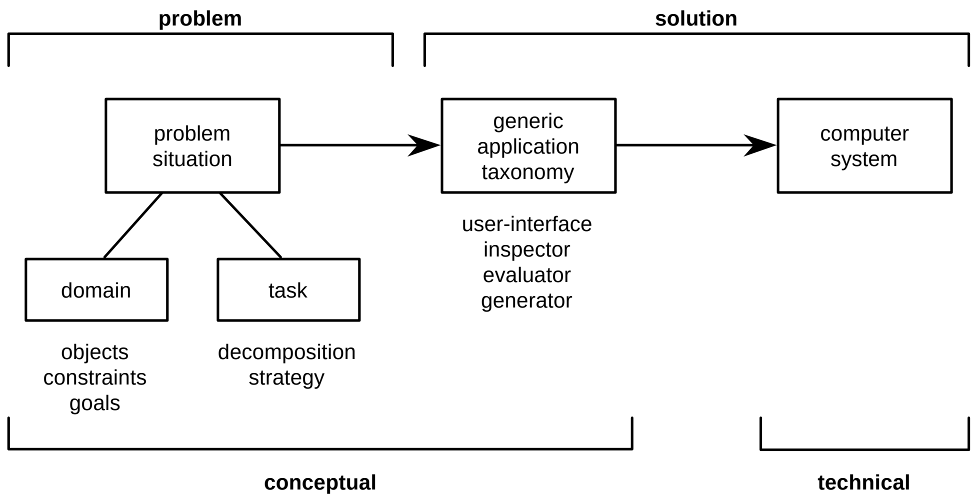


Figure 2. Models of the Scheduling Expertise Concept (Van Wezel, 1994)

The following section will discuss how these modelling techniques are applied in a reuse support system to enact reuse in the development process of planning and scheduling systems.

6.The SEC-system

In the previous section we discussed a set of models that depict the results from development processes in the planning and scheduling domain. The model types are specified in such a way that the actual compilation of these models does not have to be based solely on case specific information. Model developers can use previously built models to create models for a new case. The search for usable model components, however, requires considerable information processing. To aid the search, we develop the SEC-system that is based on the SEC-model. The SEC-system adds domain specific information to the empty concepts of analogical reasoning and faceted classification. Although our main research is focused towards a composition approach, we used a generation approach for the user-interface: the Scheduling User-interface for Prototyping and Knowledge Acquisition (Supka; Huisman et al., 1994). The remainder of this section will discuss the functional requirements and a partial implementation of the SEC-system.

The SEC-system is a reuse support system that combines analogical reasoning, model decomposition, and faceted classification with the various models in the SEC-model. Although Sibum et al. (1994) describe the overall functional requirements of the SEC-system, until now only limited functionality is implemented in a prototype.

The requirements of the SEC-system can be derived from the reuse approaches that we described. To apply analogical reasoning, we need a set of models that can be decomposed into components. These components are models themselves that can be decomposed further. Decomposition can proceed until not-decomposable primitives are encountered. The SEC-model provides such models. The object models allow decomposition by means of aggregation and abstraction mechanisms. An example of decomposition by aggregation is the aggregate object *shift*. It is a component of the aggregate object *scheduled shift*, and it can be decomposed further itself into a starting time and an ending time. Abstraction is the second form of decomposition of object models. An example is the object *employee*, which describes a subset of the properties of the object *nurse*. Decomposition can proceed until either singular objects or primitive objects are encountered.

Decomposition of other models takes place in the same way. The task structure can be decomposed in sub-tasks or abstract tasks by referring to abstract objects. The task strategy decomposes into strategy-parts and strategies that refer to abstract sub-tasks. The user-interface can be decomposed into groups of widgets and the schedule generator decomposes in the same manner as task structures and task strategies. The inspector and evaluator refer to singular or aggregate objects and hence decompose accordingly.

The thesaurus to support faceted classification is constructed in two ways. First, the objects and tasks have names and these can be put in the thesaurus. Second, to add more semantics, the perceived meaning can be added to the thesaurus, for example the strategy features that were described. The distance in the inheritance tree, i.e. the number of properties that objects have in common, can be used as a measure for closeness of words in the thesaurus. Figure 3 depicts a conceptual data-model of the SEC-system. Decomposition of models is represented by the "part-of" relationship. The "is-a" relationship models abstraction of objects. All data entities have a name, and, where appropriate, several meanings. For each entity, a separate table contains closeness factors between instances of the entity.

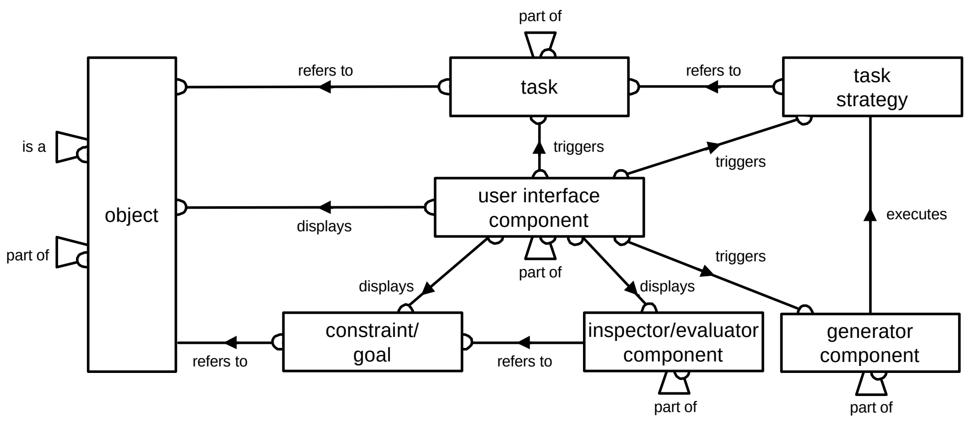
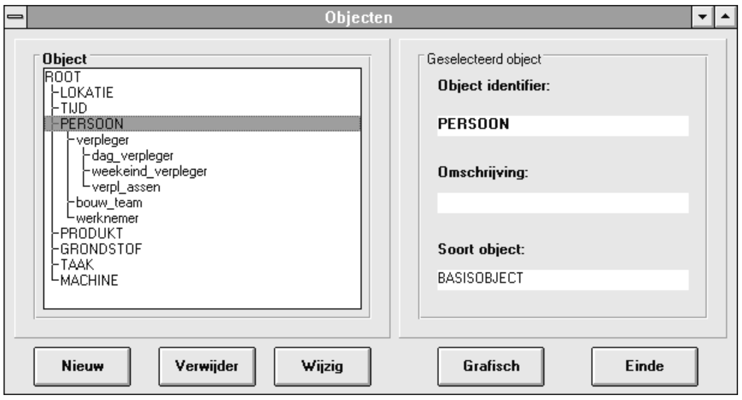


Figure 3. Data-model of SEC-system

Current implementation of the SEC-system incorporates: (1) a database structure and graphical browser for an inheritance tree of objects (Barten, 1994; Figure 4a); (2) a structure for questionnaires; (3) a questionnaire for the scheduling situation including objects and tasks; (4) a questionnaire for scheduling systems with an emphasis on staff scheduling; and (5) a method for comparing scheduling situations and scheduling systems (van Gestel, 1996; Figure 4b). The only data entities that are currently implemented are the objects and the "is-a" relations. Therefore, the SEC-system lacks the other entities and their relations.

The questionnaires are used to gather the required information in a structured way. The questions are domain dependent. Currently, the questionnaire is being extended with questions for the food processing industries. Ideally, one questionnaire should incorporate all questions that are to be asked to develop adequate scheduling support. However, to join in with the ever changing and evolving domains of reality, we adopt the composition approach. The SEC-system provides a flexible method for creating and maintaining questionnaires.

In addition, filled-in questionnaires can be compared and the SEC-system displays the similarities and differences (Figure 4b). Eventually, the SEC-system should be able to translate the results of a questionnaire to the SEC-models. This, however, is not yet implemented, and is part of the ongoing research.



Object

```
ROOT
├LOKATIE
├TIJD
├PERSOON
│ ├verpleger
│ │ ├dag_verpleger
│ │ ├weekeind_verpleger
│ │ └verpl_assen
│ ├bouw_team
│ └werknemer
├PRODUKT
├GRONDSTOF
├TAAK
└MACHINE
```

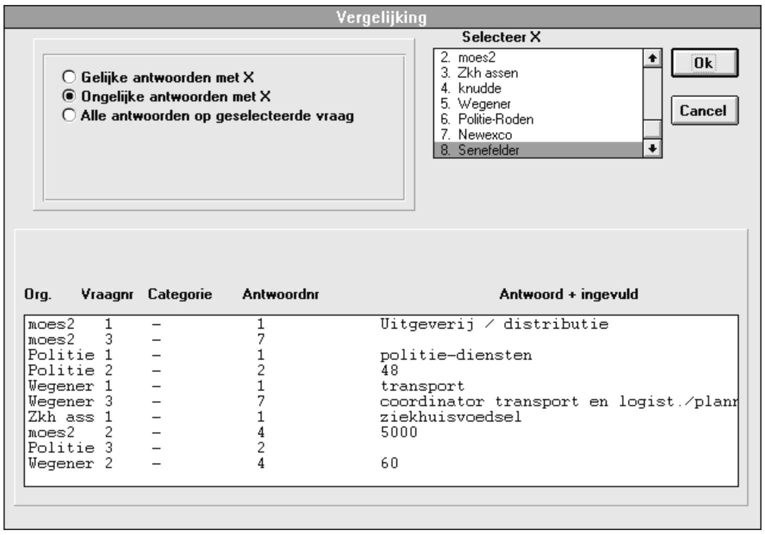
Geselecteerd object

Object identifier: PERSOON

Omschrijving:

Soort object: BASISOBJECT

Figure 4a. Object tree in the SEC-system



Gelijke antwoorden met X [✓ Ongelijke antwoorden met X] Alle antwoorden op geselecteerde vraag

Selecteer X:
2. moes2
3. Zkh assen
4. knudde
5. Wegener
6. Politie-Roden
7. Newexco
8. Senefelder

```
Org.     Vraagnr  Categorie   Antwoordnr            Antwoord + ingevuld
moes2    1        -           1            Uitgeverij / distributie
moes2    3        -           7
Politie  1        -           1            politie-diensten
Politie  2        -           2            48
Wegener  1        -           1            transport
Wegener  3        -           7            coordinator transport en logist./planr
Zkh ass  1        -           1            ziekhuisvoedsel
moes2    2        -           4            5000
Politie  3        -           2
Wegener  2        -           4            60
```

Figure 4b. Questionnaire comparison in the SEC-system

7.Conclusions and further research

There are two movements in computer system development that, although at first sight opposing, have lead to the need for reuse of conceptual and technical products. On the one hand there is a tendency to create bigger and more encompassing computer systems, on the other hand there is a demand for small intelligent systems that are really adapted to individual users. Both developments require that much time and expertise have to be invested in thorough analyses of task domains and the individual task performance of human users. With the need for more comprising and more intelligent systems, the complexity, desultoriness, and dynamics of systems increase. Nowadays, computer systems do not exist five years unaltered anymore, they are continually adapted. In addition, computer support changes the task performance. For these reasons, it is apparent that reuse of conceptual and technical components is considered and will soon be inevitable.

Instruments to realise reuse in a simple way, however, are lacking. There are no generally accepted task structures or task models, and there is no firmly established ontology or semantic/semiotic framework. We encountered these deficiencies when we tried to formulate components for reuse after building several (prototype) scheduling support systems. It turns out that computer science with regard to reuse issues is still lacking a firm fundament. It is not enough to set up a reuse approach, consisting only of database theory, programming languages, and decision support systems theory. Much more is needed, and for the time being the missing parts must come from other scientific fields.

In our research we analysed and applied methods, procedures, instruments, and approaches from several of these fields in order to deal with the emerging issue of conceptual and technical reuse. Concerning categorisation we used parts of linguistics (semantic networks and communication (Sowa, 1984)) and in relation to extended (cognitive) task analyses we worked in the tradition of cognitive psychology and cognitive science (problem solving and human information processing (Newell & Simon, 1972; Schank & Abelson, 1977)). Philosophy, especially epistemology and ontology (Quine, 1960; Goodman, 1981; Bunge, 1973)) was used to deal with "reality and modelling" discussions and semiotics – as the study of sign and sign structures – was implicitly used to compare and analyse symbols and other sign structures in modelling and representation (Jorna, 1990). We applied the mix that was the pragmatic combination of these fields to reuse discussions. Our objective is to facilitate planning and scheduling reuse in a conceptual as well as in a software modular sense; it was not our intention to be purely philosophical or psychological. In our opinion, not only the design and development of the SEC-model and the SEC-system profited from it, but also the realisation of several of the – sometimes prototypical – scheduling support systems we built (Mietus, 1994; Van Wezel et al., 1996).

In summary, the Scheduling Expertise Concept is a user-centred and reuse-enabling scheduling support system development method. The SEC-models combine ideas from generic reuse-approaches, decision support theory, and ideas from various scientific areas that engage in planning or scheduling, in an integrated planning concept. Several existing scheduling support approaches are examined from the perspective of reuse with respect to problem representation, domain, and system aspects to give content to the SEC-models. The SEC-models are partially implemented in the SEC system, that offers assistance with system development by searching existing components and indicating what modifications or additions are necessary to these components.

Further research will extend the SEC-system to incorporate the specified functional requirements. In addition, the Scheduling Expertise Concept will be elaborated with respect to both the structure and the content of the models. This will be realised by ongoing theoretical research as well as application of the framework in practice. This incorporates aspects of the organisation of planning, e.g., multiple planning with several interrelated and dependent plannings, and domain specific analyses, e.g., in the food processing industries. In combination, the SEC-models and SEC-system will provide both faster development processes and supporting systems with a higher quality.

8. Acknowledgements

Various people participated in research projects that relate to the SEC-model. In particular Simon Sibum and Erik Huisman need to be mentioned as cofounders of the SEC-system. Erik Barten and Hans van Gestel created the current prototype.

9. References

Anthony, R.N. (1965). *Planning and Control Systems. A framework for Analysis*. Boston: Harvard.

Bakker, J. (1995). *Classificatie en diagnose voor scheduling situaties*. Groningen: Internal report, University of Groningen.

Barten, E. (1994). *Prototype SEC-systeem*. Groningen: Internal report, University of Groningen.

Beck, H. & A. Tate (1996). Open planning, scheduling and constraint management architectures. *British Telecommunication's Technical Journal, Special Issue on Resource Management*, 1995

Biggerstaff, T.J. & A. Richter (1987). Reusability framework, assessment, and directions. *IEEE Software*, vol. 4, no. 2.

BiggerStaff, T.J. & A.J. Perlis (1989). *Software Reusability*. Reading, MA: ACM Press.

Breuker, J. & W. van der Velde (Eds.) (1994). *CommonKADS library for expertise modelling: reusable problem solving components.* Amsterdam: IOS Press.

Bunge, M. (1973). *Method, model and matter*. Dordrecht: Reidel.

Card, S.K., T.P. Moran & A. Newell (1983). *The psychology of human-computer interaction*. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Clancey, W.J. (1985). Heuristic classification. *Artificial Intelligence*, 27, p. 289-350.

Eberts, R.E. (1994). *User Interface Design*. New Jersey: Prentice Hall.

Ehlers, E.M. & E. van Rensburg (1996). An object-oriented manufacturing scheduling approach. *IEEE Transactions on systems, man, and cybernetics - part A: systems and humans*, vol. 26, no. 1, p. 17-25.

Fillmore, C.J. (1968). The case for case. In: E. Back & R.T. Harms (Eds.), *Universals in Linguistic Theory*. New York: Holt, Rinehart & Winston.

Fox, J., Gierl, C., Johns, N. & Rahmanzadeh, A. (1996). PROforma: Draft Language Overview (version 1.0); A contribution to the ACTION initiative of the European Union Healthcare Telemathics Programme. Internal Report Imperial Cancer Research Funds Laboratories, London.

Freeman, P.A. (1987). Conceptual analysis of the Draco Approach to Constructing Software Systems, *IEEE Transactions on Software Engineering*, Vol. SE-13, 7, p. 830-844.

Fukunaga, A. & G. Rabideau (1997). Towards an Application Framework for Automated Planning and Scheduling. In: *Proceedings of IEEE Aerospace Conference,* Snowmass, CO, 1997.

Goodman, N. (1981). *Languages of Art*. Brighton, Sussex: The Harvester Press (First Edition 1968).

Henke, A. (1994). Scheduling space shuttle missions: using object-oriented techniques. *AI Expert*, march 1994, pp 16-24.

Huisman, E., R.J. Jorna & W. van Wezel (1994). *Algehele structuur voor het SEC, in het bijzonder generator, controleur, evaluator, randvoorwaarden en doelfuncties.* Groningen: Internal report, University of Groningen.

Jorna, R.J. (1990). *Knowledge Representation and Symbols in the Mind*. Stauffenbrug Verlag: Tubingen.

Jorna, R.J. & W. van Wezel (1996). Objects and the World Metaphor: a Semiotic Engineering Approach. In: Nöth, W. (ed.). *Semiotics of the Media*. New York: Springer-Verlag.

Jorna, R.J., H.W.M. Gazendam, H.C. Heesen & W.M.C. van Wezel (1996). *Plannen en roosteren. Taakgericht analyseren, ontwerpen en ondersteunen.* Leiderdorp: Lansa Publishing B.V.

Kwon, O.B. & S.J. Park (1996). RMT: A modeling support system for model reuse. *Decision Support Systems : the international journal*, 16, p. 131-153.

Liang, T.P. (1993). Analogical reasoning and case-based learning in model management systems. *Decision Support Systems : the international journal*, 10, p. 137-160.

Liang, T.P. & Konsynski, B.R. (1993). Modeling by analogy: Use of analogical reasoning in model management systems. *Decision Support Systems : the international journal*, 9(1), 113-126

Liu, B. (1993). Problem acquisition in scheduling domains. *Expert Systems with Applications, vol. 6,* pp. 257-265.

Mietus, D.M. (1994). *Understanding planning for effective decision support*. Groningen: Ph.D. thesis, University of Groningen.

Newell, A. & Simon, H.A. (1972). *Human Problem Solving*. Englewood Cliffs, New Jersey: Prentice-Hall, Inc.

Pillutla, S.N. & B.N. Nag (1996). Object-oriented model construction in production scheduling decisions. *Decision Support Systems : the international journal*, 18, p. 357-375.

Prieto-Díaz, R. (1991). Implementing Faceted Classification for Software Reuse. *Communications of the ACM*, 34, 5, p. 89-97.

Prietula, M.J., W. Hsu & P.S. Ow (1994). MacMerl: Mixed-Initiative Scheduling with Coincident Problem Spaces. In: Zweben, M. & Fox, M.S. *Intelligent Scheduling*. San Francisco: Morgan Kaufman.

Quine, W.V.O. (1960). *Word and Object.* Cambridge, MA: MIT Press.

Ramesh, M. & Raghav Rao, H. (1994). Software reuse: Issues and an example. *Decision Support Systems : the international journal*, 12, p. 57-77.

Riesbeck, C.K. & Schank, R.C. (1989). *Inside case-based reasoning*. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Russell, S. & Norvig, P. (1995). *Artificial intelligence, a modern approach.* Englewood Cliffs, New Jersey: Prentice-Hall.

Schank, R. & R. Abelson (1977). *Scripts, Plans, Goals and Understanding*. Hillsdale: Erlbaum Associates.

Schreiber, A.Th. (1993). Applying KADS to the Sisyphus Domain. In: Schreiber, A.Th., Wielinga, B. J., & Breuker, J. A. (Eds). *KADS: a Principled Approach to Knowledge Engineering, volume 11 of Knowledge-Based Systems Book Series*. Academic Press, London.

Sibum, S., E. Barten, E.H. Huisman & W. van Wezel (1994). *Het SEC-systeem: defnitiestudie.* Intern rapport Diskus-C, Rijksuniversiteit Groningen.

Sikkel, K. & J.C. van Vliet (1988). Kan software langer mee? *Informatie*, 7/8, p. 464-483.

Smith, S.F. & O. Lassila (1994). Toward the Development of Flexible Mixed-Initiative Scheduling Tools. In: *Proceedings ARPA-Rome Laboratory planning initiative workshop*, Tucson, AZ.

Smith, S.F, O. Lassila & M. Becker (1996). Configurable, Mixed-Initiative Systems for Planning and Scheduling. In: Tate, A. (Ed.). *Advanced Planning Technology*. Menlo Park, CA: AAAI Press.

Smith, S.F. & M. Becker (1997). "An Ontology for Constructing Scheduling Systems", *Working Notes of 1997 AAAI Symposium on Ontological Engineering.* Stanford, CA: AAAI Press.

Sowa, J.F. (1984). *Conceptual Structures.* Reading, MA: Addison Wesley.

Sundin, U. (1994). Assignment and Scheduling. In: Breuker, J. & W. van der Velde (Eds.) *CommonKADS library for expertise modelling: reusable problem solving components*. Amsterdam: IOS Press.

Tate, A. (1993). The Emergence of “Standard” Planning and Scheduling System Components - Open Planning and Scheduling Architectures. In: *European Workshop on Planning (EWSP ’93)*.

van Genuchten, M., T. Bemelmans & F.J. Heemstra (1993). De software-fabriek: beheersing van ontwikkeling, onderhoud en gebruik. *Informatie*, 35, 4, p. 258-267.

van Gestel, J.P.W. (1996). *Prototype SEC-systeem. Het vervolg*. Groningen: Internal report, University of Groningen.

van Putten, J., N. Scharenborg & A. Woerlee (1993). A Generic User Interface Constructor for Planning and Scheduling Applications. *Proceedings of the HCI'93 Conference on People and Computers VIII*.

van Wezel, W.M.C. (1994). D*e SEC-methodiek; ontwikkelen van scheduling-applicaties*. Groningen: Internal report, University of Groningen.

van Wezel, W. & D.P. van Donk (1996). Scheduling in Food Processing Industries: Preliminary Findings of a Task Oriented Approach. In: Bertrand, Jafari, Fransoo & Rutten (1996). *Second International Conference on Computer Integrated Manufacturing in the Process Industries – Proceedings*, pp. 545-557

van Wezel, W., R.J. Jorna & D. Mietus (1996). Scheduling in a generic perspective. *International Journal of Expert Systems; research and applications*, 3 (9), pp. 357-381.

Veloso, M.M. (1996). Towards Mixed-Initiative Rationale-Supported Planning. In: Tate, A. (Ed.). *Advanced Planning Technology*. Menlo Park, CA: AAAI Press.

Verbraeck, A. (1991). *Developing an Adaptive Scheduling Support Environment.* Delft: Ph.D. Thesis, University of Delft.

Wennink, M. (1995). *Algorithmic Support for Automated Planning Boards*., Eindhoven: Ph.D. thesis, Technical University of Eindhoven.

Wennink, M. & M. Savelsbergh (1996). Towards a planning board generator. *Decision Support Systems : the international journal*, 17(3), p 199-226.

Woerlee, A.P. (1991). *Decision Support Systems for Production Scheduling*. Rotterdam: Ph.D. thesis, Erasmus University Rotterdam.

Wolf, G. (1994). Schedule management: an object-oriented approach. *Decision Support Systems : the international journal*, 11, pp. 373-388.

Zweben, M. & M.S. Fox (1994). *Intelligent Scheduling*. San Francisco: Morgan Kaufman.