International Journal of Computer & Information Science, Vol. 2, No. 3, September 2001

A Task Analysis and Design Framework for Management Systems and Decision Support Systems

Michael Herczeg
University of Luebeck, Institute for Multimedia and Interactive Systems
http://www.imis.uni-luebeck.de
D-23568 Luebeck, Germany
EMail: michael.herczeg@acm.org

Abstract:

This contribution describes a framework with organizational and operating concepts for the management, i.e. supervision and control of complex dynamic systems (processes), like networks, factories, power plants or transport systems. This kind of system operation has also been called supervisory control [10], [15]. These applications are characterized by a complex allocation of tasks between several users (operators) and machine agents (system functions).

For systems to be designed, the concepts of the framework have to be transformed into application-specific terms according to the new management application. This step will create an application-oriented framework to design new management systems in a way that they are able fulfil the operational and human requirements to meet the business goals. For existing systems to be analysed, the entities have to be transformed into the typical application terms in respect to these management applications. This will provide an application-oriented framework for an analysis to evaluate the system, identify problems, improve and optimize the system.

This framework has been developed mainly in the area of telecommunication network management and has been used to design telecommunication management systems for digital broadband networks. The concepts are expected to applicable in other application areas as well.

Keywords:

Decision Support, Supervisory Control, User Interface, Task Analysis, System Design Work Flow, Work Scenario, Task, Role, Tool, Managed Object, User Qualification.

1. Management Systems

A management system is a work system [1]. Management systems deal with the management of the resources of a dynamic system, called the **process**. The work system consists of the following abstract entities (see Figure 1. Framework Entities):

Managed Objects: resources of the dynamic system to be managed represented by active information elements; the set of all managed objects is called the management information base (MIB)

Tasks: situations, where the system shall be supervised or a system state shall be changed into a defined goal state by the execution of a work procedure

Roles: interrelated organisational entities in the management system executing defined tasks

Agents: human operators or machine functions representing roles; human operators (users) are defined by their qualifications

Tools: support systems for the execution of tasks

The abstract entities described above are the concept-layer (a kind of meta-layer) of any management system. They build the foundation of the framework to be defined.

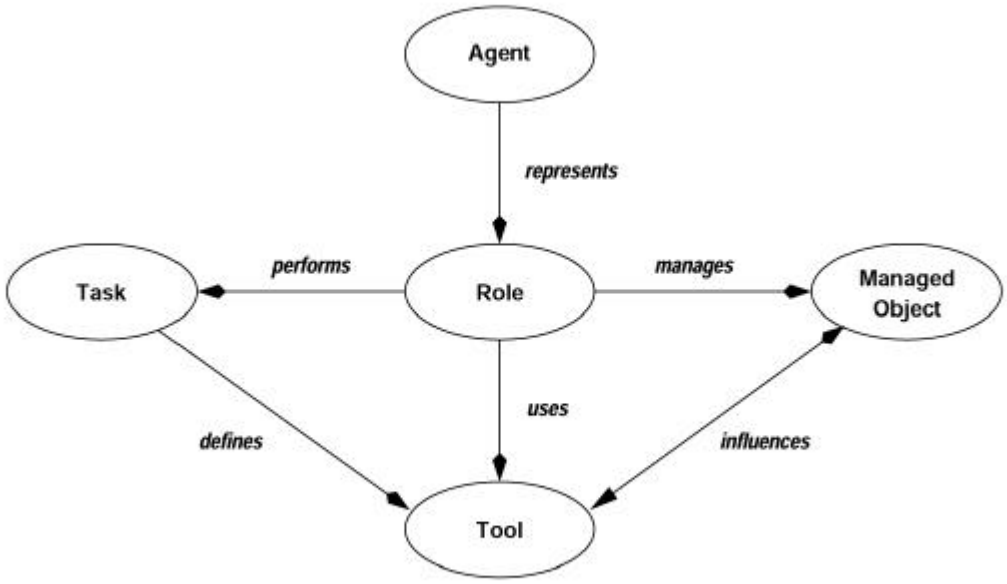

Figure 1. Framework Entities

2. Design Layers

The following outline is based on the concept of object-oriented system design.

Each of the conceptual entities of a management system have to be refined into several classes. Managed Objects will be described as specific Managed Object Classes, Tasks as Task Classes, etc. These classes will contain application specific knowledge about the management system and the system to be managed. For a real implemented management system, all of these classes have to be instantiated into problem-domain specific objects by assigning real process resources (managed objects).

As a result of this classification and instantiation process, there are three layers of abstraction in this framework for operating concepts, the framework layer, the class layer and the instance layer.

2.1 Framework Layer

The framework layer consists of the application independent framework entities like *managed object*, *task*, *role*, *agent*, and *tool*. These entities define the framework itself.

Examples of entities:

Managed Objects: described by the abstract entity *managed-object*
Tasks: described by the abstract entity *task*
Roles: described by the abstract entity *role*
Agents: described by the abstract entity *agent*
Tools: described by the abtract entity *tool*

2.2 Class Layer

The class layer consists of application specific classes for managed objects, tasks, roles, agents, and tools.

Examples of classes:

Managed Objects: a managed object class, like *equipment*
Tasks: a task class, like *put-board-into-service*
Roles: a class of a role, like *network-operator*
Agents: an agent class, like *technical-staff*
Tools: a tool class, like *alarm-table*

2.3 Instance Layer

The instance layer consists of application related instantiations of the task classes, role classes, agent classes and tool classes mainly by assigning specific managed object instances to attributes of these classes.

Examples of instances:

Managed Objects: an instance of *equipment*, like *board-138*

Tasks: an instance of *put-board-into-service*, like *put-board-138-into-service*

Roles: an instance of *network-operator*, like *network-operator-for-region-1*

Agents: an instance of *technical-staff*, like *John-Miller*

Tools: an instance of *alarm-table*, like *equipment-alarm-table*

To design or analyse a management system all of these layers have to be discussed, structured and documented. This contribution will focus on the framework layer.

In the case of an analysis of an existing management system the class layer has to be reconstructed using the implemented instance layer of the existing management system. In the case of the design of a new management system the class layer has to be constructed based on the framework layer before the instance layer can be set up.

3. Analysis and Design Procedure

The following procedure outlines steps in the design of a management system together with its operating concepts. The order of the steps is a proposal for a situation when everything is designed from scratch. If there are already fragments of a system or even a working system to be optimized or reengineered, some steps may already be completed. In such a case it should be analysed at least, whether the existent situation is a really acceptable satisfying solution.

1. ***Task-Design***: define the task classes and the task structure
2. ***Role-Design***: define the role classes
3. ***Task-Allocation***: assign task classes to role classes
4. ***Qualification-Definition***: define the operator classes (may already be defined)
5. ***Qualification-Allocation***: assign operator classes to role classes
6. ***Tool-Design***: design the tool classes in respect to the task, role and operator classes
7. ***Tool-Allocation***: assign the tool classes to the task classes
8. ***Resource-Assignment***: instantiate task classes with managed object instances
9. ***Workflow-Design***: define, analyse and optimize the workflow between role classes
10. ***Role-Assignment***: instantiate role classes for task instances
11. ***User-Classification***: assign operators (real persons) to operator classes
12. ***User-Assignment***: assign operators to role instances

This procedure will be performed in an incremental way. Some concepts will be modelled in a first step roughly and will be refined in a later iteration. The application world with its application objects (managed objects) are expected to be defined prior to this analysis and design procedure.

4. Managed Objects

System resources are described as managed objects. They represent the information space of the application, i.e. the process elements to be managed.

Managed objects will often be named as **application objects** or **domain objects** because they will usually be described in terms of the application world to be managed.

Managed object instances are the entities of systems to be managed. The set of managed object instances is called the **management information base** (MIB) of a management application. The managed object instances are defined by **managed object classes** describing the attributes and methods of their instances. The system of managed object classes is called the **management information model** of a management application.

To manage a system state operations will be applied to these resources. The set of states of all system resources define the **system state**.

The analysis and design of the management information model is outside this scope of the design of management systems and will therefore only be touched in this contribution. In the area of telecommunication network management systems it has been described in detail by the ITU in the X.700-series [3], [4], [5], [6].

5. Tasks

Tasks are the result of following business goals. Tasks result in activities to reach these goals. Only business oriented tasks will result in profit for the organisation managing a process. These productive tasks appear in a direct relationship with the management goals and will be called **external tasks**. Other tasks cope with artificial goals resulting from the implementation of the process and the tools to supervise and control the process. These tasks will correspondingly be called **internal tasks**. Many of these internal tasks will be non-productive, i.e. are not directly related to external tasks and should be minimized as far as possible [1], [9].

A detailed task analysis is necessary to be able to design a work system in way that concentrates on external tasks and minimizes the creation of non-productive internal tasks. However, it has to be ensured that all emerging user activities related to tasks can be handled in an adequate way, i.e. human operators can handle the tasks in an efficient and correct way.

Some tasks can be performed by machine agents (functions) instead of human agents (operators). The task analysis will give important hints about **task allocation** to human or machine operators (automatic operating functions) [16].

Tasks are usually defined in respect to system resources (managed objects). Task classes are defined in respect to managed object classes and task instances are defined to deal with managed object instances.

5.1 Task Classes

The following characteristics have to be analysed for each task class:

Purpose: reason and goal of the task
clarifies whether the task is an external or internal task

Content: description of task content, mainly the description of the operational procedures applied to managed objects (at this level of description represented by the managed object classes)
provides an understanding of the task for system designers and operators

Subtasks: decomposition of the task into subtasks
provides the task structure

Supertasks: tasks the described task is part of
will only be available if task is a subtask of some other tasks

Input: input data needed to perform the task
defines the flow of data upstream of the workflow

Pre-State: system state before task can be performed
allows to check automatically by the tools whether they may be applied

Output: output data created by the execution of the task
defines the flow of data downstream of the workflow

Post-State: system state after task has been performed
allows to check automatically by tools whether the application of the tool has been successful

Frequency: frequency of task in the spectrum of all tasks
a high frequency shows that more efficient tools are needed to perform the task

Repetitivity: direct repetitions of the task
a high repetitivity shows that tools with repeat (again) functions are needed to perform the task efficiently

Urgency: dynamic priority of task in relation to other tasks that shall be performed at a specific time
shows whether the task needs high attention of the operator or should be allocated as a machine function performed automatically when necessary

Severity: static priority (importance) of task or situation in the spectrum of all tasks
shows whether the task is of high importance for the work result and needs high attention of the operator and tools should check pre-state and post-state

Security: requirements for the authorized execution of the task
security is strongly related to roles and their access rules

Safety: requirements for the correct execution of the task
safety implies that the execution of the task should be accompanied by automatic checks or system support in respect to activity order and state changes

Timing: time requirements for the execution of the task (e.g. duration)
high timing requirements show that high concentration of operator and efficient or highly automated real-time tools are needed

Tools: reference to tool classes
to specify the tools which shall be used to perform the task

Agent: reference to agent class
to specify the knowledge and skills of human or machine operators required to perform the task

What has been described above are task classes. These task classes may be instantiated with real process resources (managed object instances) that shall be managed by the task.

The task classes described can be structured into a task class inheritance hierarchy. This allows to factor out general characteristics into superclasses and to represent similarity of tasks. This inheritance structure is different to the task structure, which will be described in the next section.

The task description gives lots of hints about the allocation of tasks. Especially time and safety related attributes will imply to a high extent whether the task can be performed by more or less skilled operators or has to be allocated to machine functions to be performed properly.

5.2 Task Structures

Tasks will often be refined in a hierarchical way (see Figure 2: Task Structure). The hierarchy represents the breakdown of each task into a structure of subtasks.

Tasks should be substructured (refined) into lower level tasks (subtasks) if one layer of tasks will not be sufficient to cope with the complexity of an application area. This management task substructure will usually lead to a directed decomposition graph instead of a simple hierarchy since some lower level tasks will be identical for different higher level tasks.

There are several ways of combining the subtasks to realize the supertask. Theses types of combination can be viewed as control flow operators:

sequence: the task is done by performing the predefined sequence of subtasks
tasks which are performed in a predefined sequence will be called **task-phases**

parallel: the task is done by performing the subtasks in parallel
tasks which are executed and controlled in parallel will be called **task-threads**

choice: the task is done by performing the subtasks in any sequence
tasks which may be selected out of a set of tasks will be called **task-options**

In any situation the execution of a task may be dependant on certain decisions based on the current state of managed objects and the current goals. This influences whether there will be none, one or several tasks that can be executed within any of the subtask combination concepts described above. Depending on the availability of explicit criteria about the selection of the next tasks, tasks may be more or less easily allocated to a machine in the sense of automatic execution of several tasks. The complexity of the decision about the next task characterizes the **openness of the work** to be done.

Tasks do not need to be substructured into subtasks, i.e. some tasks may have subtasks whereas others are defined without subtasks. The process of refinement may stop at any task, when this task shall be viewed as an elementary task at the moment for some reason, e.g.:

- there is a system function which will support the task directly
- the task will be performed by some basic manual activity
- the task will be performed automatically by some system function

- the task is of no further interest in the analysis and design process

As the task analysis will be an iterative process, the refinement may be stopped and continued when new information is available throughout the analysis and design process.

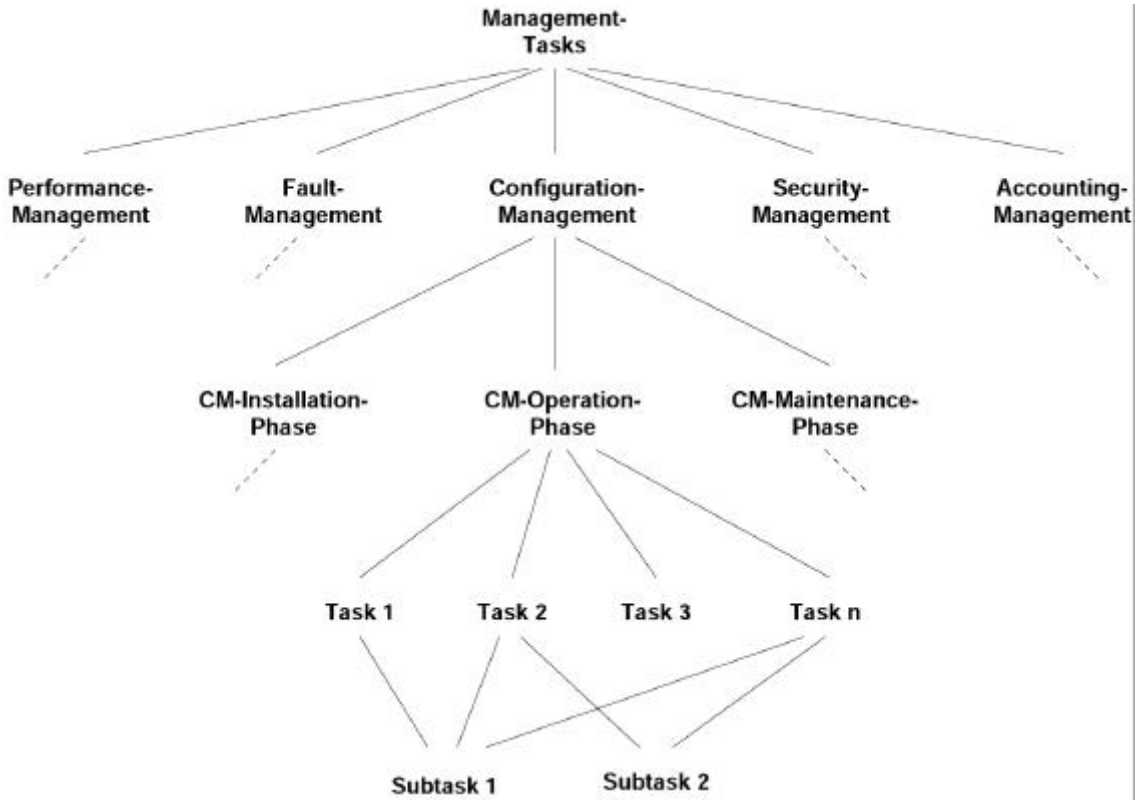

Figure 2: Task Structure

5.3 Task and System Modelling

The definition and documentation of the task structure and the task descriptions may be difficult and time consuming activities on the road to clear and efficient operating concepts. Meanwhile, there are certain experiences how it may be done without getting lost in the complexity and volume of tasks to be analysed and described.

The first two concepts (**management areas** and **management phases**) lead to the high level work structure of a task hierarchy and the second two concepts (**management abstractions** and **management decompositions**) guide the definition of the lower level task structure with strong relations to the resources of the process being managed and the system functions to be provided (see Figure 3: Task Model).

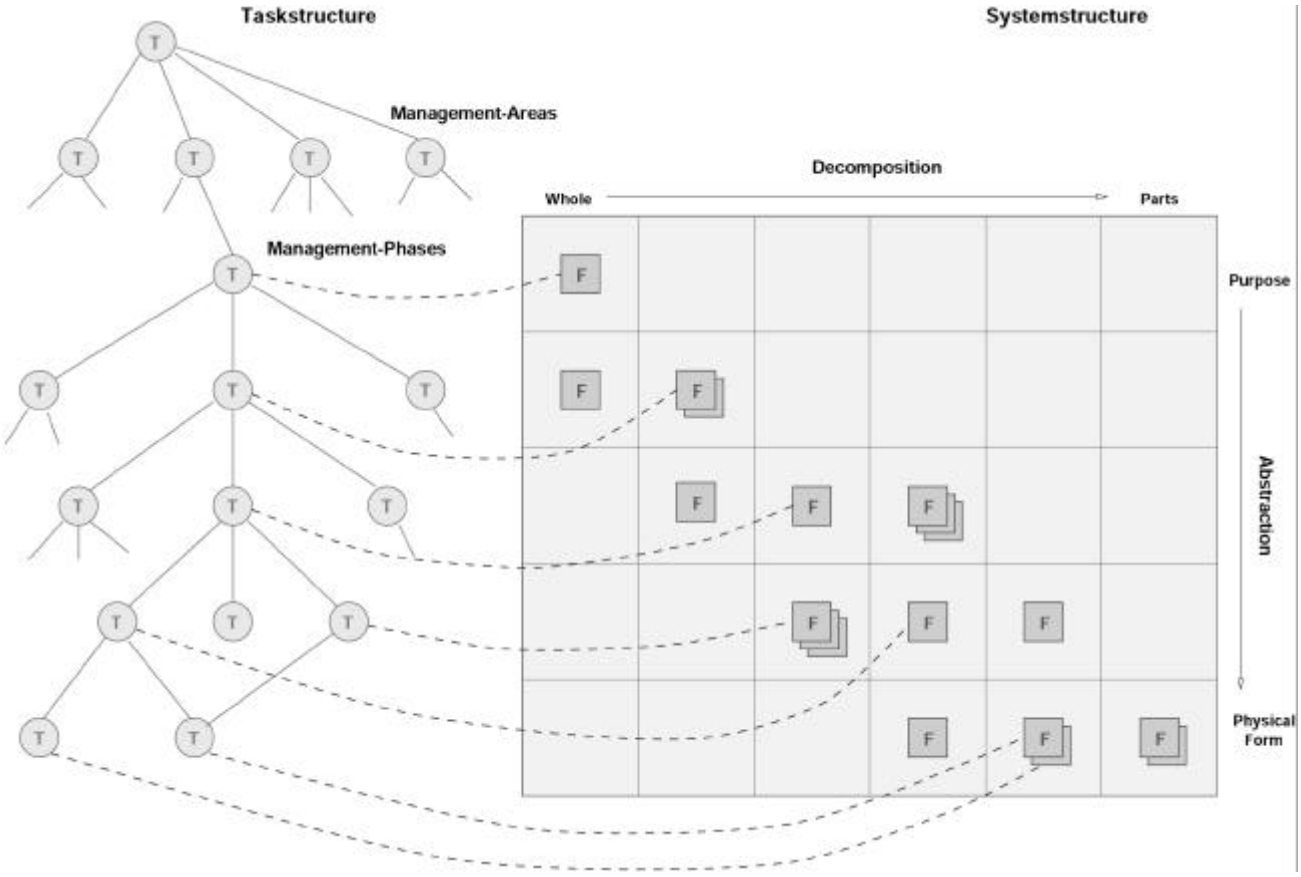

Figure 3: Task Model

5.4 Management Areas

Management areas define global sections for tasks to be distributed over high level organisational structures. Different management areas will usually be associated with different roles or even different departments of an organisation. Therefore, management areas stem from high level responsibilities within an organisation. They will usually describe specific problem areas in respect to management tasks.

For example in many technical systems the following management areas have been widely used [7]:

- Configuration-Management
- Fault-Management
- Performance-Management
- Security-Management
- Accounting-Management

5.5 Management Phases

Activities within management areas will often be structured into several main operational phases. These phases will describe activities over certain time frames. The activities in different phases can be performed by one or several roles. This determines mainly whether there will exist a workflow through several roles or groups of roles of an organisation.

Typical examples of phases in the management of technical systems will be:

1. installation
2. operation
3. maintenance

5.6 Management Abstractions

Activities can be organized in several layers along an abstraction hierarchy, starting from the highest layer dealing with the intention and purpose of the work and ending on the physical layer with all of the possible activities on the process resources. Specific management operations and presentation of the system

resources (managed objects) will appear on each of this system layers. The lower the layer the more physical or naturalistic the presented or manipulated attributes of the managed object will be. The higher the layer the more logical or purpose-oriented attributes will be shown and manipulated.

Management of a system should primarily be focused on the layer of the purpose of the system. Only in cases where a detailed analysis or a fine-grained configuration has to take place, the lower layers of the system will be visited and manipulated down to the layer where the problem at hand can be solved completely or the task execution will be aborted because there is no solution.

In the context of process supervision and control a multi-level abstraction hierarchy has been described by Rasmussen and others [12], [13], [14] (see Figure 4: Abstraction Levels (according to Rasmussen)):

Functional Purpose: production flow models, control system objectives

Abstract Function: causal structure, mass, energy and information flow topology

Generic Function: standard functions and processes, control loops

Physical Function: electrical, mechanical, chemical processes of components and equipment

Physical Form: physical appearance and anatomy, material and form, locations

This structure provides a natural way to distinguish between external tasks and internal tasks. To make it clear, only tasks on the level of the functional purpose are external tasks. All other layers create implementation dependant and therefore internal tasks which may create non-productive additional burden to the operation of the system. All of these lower level tasks should be allocated to machine functions if possible and safe.

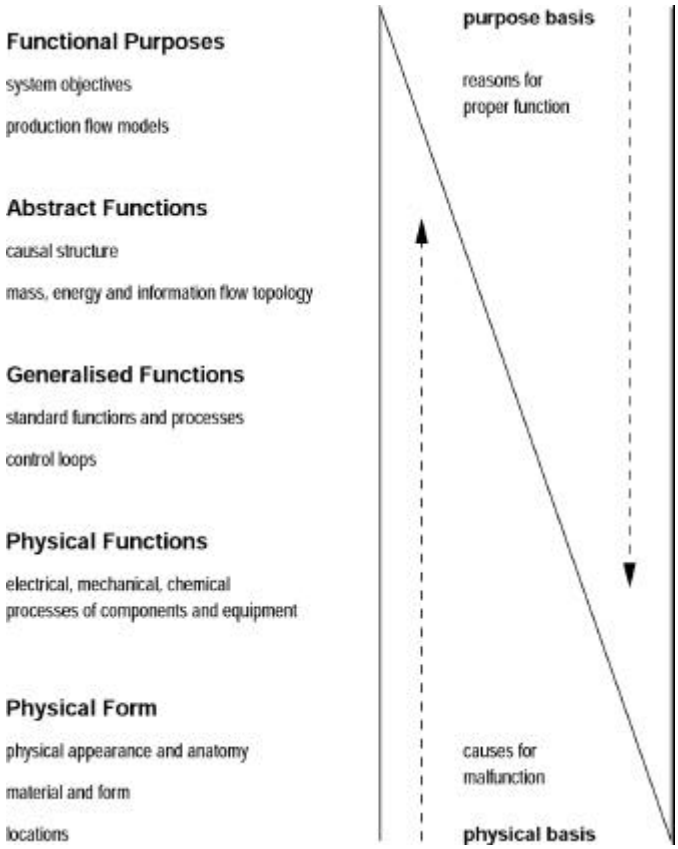

Figure 4: Abstraction Levels (according to Rasmussen)

5.7 Management Decompositions

As a second dimension orthogonal to the abstraction dimension there is the decomposition of the system leading **from the whole to the parts** of a system resource. To structure the system resources into part hierarchies is a natural way of modelling, understanding, and handling large and complex systems. The decomposition structure may be used to focus on parts of the whole system and a hierarchy of decomposition helps to structure the whole system into task-oriented substructures.

It should be noticed that this decomposition dimension is not the same as the abstraction dimension since the decomposition may be defined within each of the abstraction layers. The whole and the parts of a resource can be shown on the same level of abstraction. Nevertheless, decomposition is a kind of structural abstraction concept adressing mainly managed objects.

6. Roles

The set of interrelated roles is analysed in an organisational analysis. Roles are the placeholders for operators (end users) and are describing their function in the work system.

Roles will generally be grouped into hierarchical structures creating organisational entities like teams, departments and divisions.

Roles will be defined as role classes. Several role instances can be created for one role class. Each of these roles instances will perform its specific task instances of the same task class (i.e. the role instances will perform the same tasks applied to different managed objects).

6.1 Role Classes

The analysis of roles has to clarify the following attributes of roles:

Description: description of the roles function within the organisation

Substitute: role to take over tasks in case of role is out of operation

Tasks: references to the task classes which have to be performed by the roles
the tasks to be performed by a role imply the necessary qualifications of an agent representing a role

Pre-Roles: roles to be active before this role will be active

Input: information needed by a role to perform its tasks

Post-Roles: roles to be activated after this role has been active

Output: information created by a role when performing its tasks

Communication: which type of communication media is used between interacting roles to pass output to be used as input by another role
tells about the media used to receive input and deliver output

Tools: reference to tool classes, needed by a role to perform its tasks
the role's tools are defined indirectly by the set of all tools needed by the role to perform its tasks and must not be documented explicitly; this gives important ideas about consistency of tools and integration of tools to higher level tools for a role

Agent: reference to agent class
to specify the knowledge and skills of human or machine operators required to represent the role

The task-related characteristics (Input, Output, Tools) will usually be defined in the task analysis and may be summarized in the organisational analysis related to the roles.

6.2 Task Assignment to Roles

In the process of the task allocation task classes have to be assigned to the role classes. After this assignment role classes will usually be instantiated with different task instances, to manage different system resources (managed object instances). In certain management systems it will be comfortable to instantiate each role class once, leading to exactly one role instance per role class. This two-step assignment on the class and the instance layer has to make sure that

- each task class is assigned to at least one role class,
- each task instance is assigned to at least one role instance,
- several role instances which will manage the same task instances have to apply operational rules to avoid activity conflicts by more than one operator managing the same system resource at the same time and
- security and authorization concepts have to ensure that role instances will only be able to manage the resources defined by their task assignment.

Assignment of task classes may take place in an efficient and easy to manage way, by making use of the task structure (see Figure 5: Example of a Task Assignment to Role Classes).

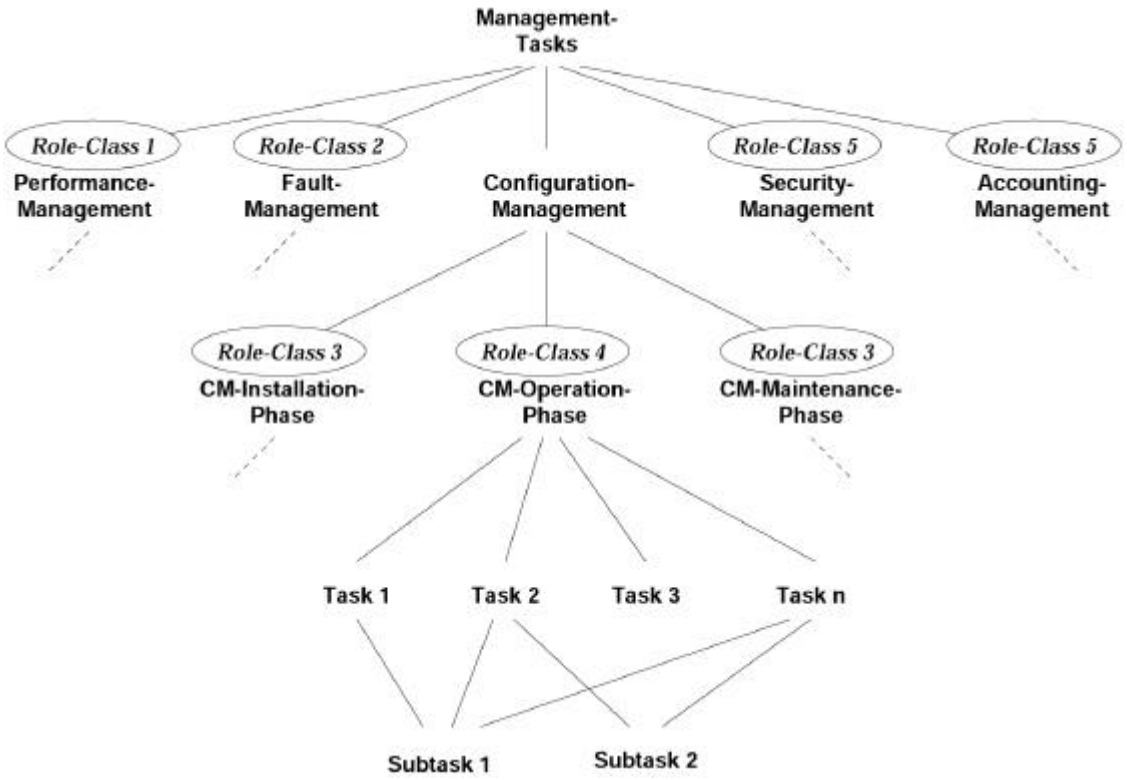

Figure 5: Example of a Task Assignment to Role Classes

The assignment of tasks may be defined in a flexible and dynamic way. This creates some burden on the organisation but allows to optimize the management system in respect to operator availability and competence.

It has to be emphasized that the assignment of tasks has to take human factors of workplace design into account [1]. In respect to task assignment it has to be ensured that tasks will be interesting and demanding instead of monotonous and primitive from an operator's point of view. As a consequence, the assignment of task classes to role classes and the assignment of operator classes to role classes are highly interrelated and have to be iterated until a satisfying constellation has been found.

6.3 Workflow between Roles

Roles interact with other roles in many ways:

- a role creates output which is needed as input for another role
- a role performs a task and creates a system state which is needed by another role
- a role activates another role by sending an activation event

These interactions can be described by workflow relations between role classes. In the instantiated roles these relations will be used to exchange data and control by certain communication methods.

Roles interact with the resources of the process:

- managed objects are manipulated by roles through the execution of tasks using tools
- roles receive notifications from the process indicating changes (**events**) in the system

By these interactions a dependency network of managed objects, roles, tasks and tools is created (see Figure 6: Workflow). As a result, tasks are dynamically chained and brought into a sequence, starting with some initial tasks and usually ending with some final tasks or principally running forever. This flow of data, events and control between classes and their execution of tasks is called **workflow**. Specific role sequences from a start to an end state are called **routes**. Routes represent higher order tasks being performed by a set of roles, sometimes called a team.

In an organisation consisting of many roles, independent tasks will usually be performed in parallel by these roles. It is an important optimization problem to maximize the grade of parallelisation in the organisation. However, the following workflow problems can be observed or anticipated:

- a role is blocked by waiting for input from one or more other roles
- a role is blocked since the system state does not allow to perform tasks by this role
- a role is not able to perform some task as a result of tool problems
- a role is not executing some task because it did not receive an activation from another role or from the process
- a role needs more time than planned to perform a task
- a role needs less time than planned to perform a task (the role capacity is not used effectively)
- a role has an input overflow (it is not able to perform tasks as fast as input data is received)
- a role has an output overflow (it is not able to get rid of produced output since the output queue is full)

Operating concepts have to be designed to avoid these problems or to provide working procedures to solve them.

6.4 Workflow Scenarios

A **worksystem** can be defined by a set of routes describing the characteristic task sequences. Routes may be described by **workflow scenarios** showing the workflow through several role classes and the usage of tools supporting the execution of the tasks. The roles have to be displayed in certain cases as role instances since it is possible that a scenario will show several instances of the same role.

Work scenarios may be described in textual form showing lists of tasks or in graphical form displaying the dependency network of roles, tasks and tools (see Figure 7: Example of a Workflow Scenario). Numbered arrows denote the sequence of tasks. Scenario graphs are directly derived from Figure 6 by dropping the presentation of managed objects. A further simplification can be reached by dropping the presentation of tools as well. To show the tools may be helpful for the design of the tools, since it will be visible which tools have to be used in a work context simultaneously.

It will usually be impossible to show all scenarios which may occur in real worksystem. In this case it will be helpful to describe the most important **key scenarios**, with the following characteristics:

- many roles involved
- complex interrelations between roles
- substantial part of the work time spent
- most important or most sensitive tasks being performed
- most complex tooling situation

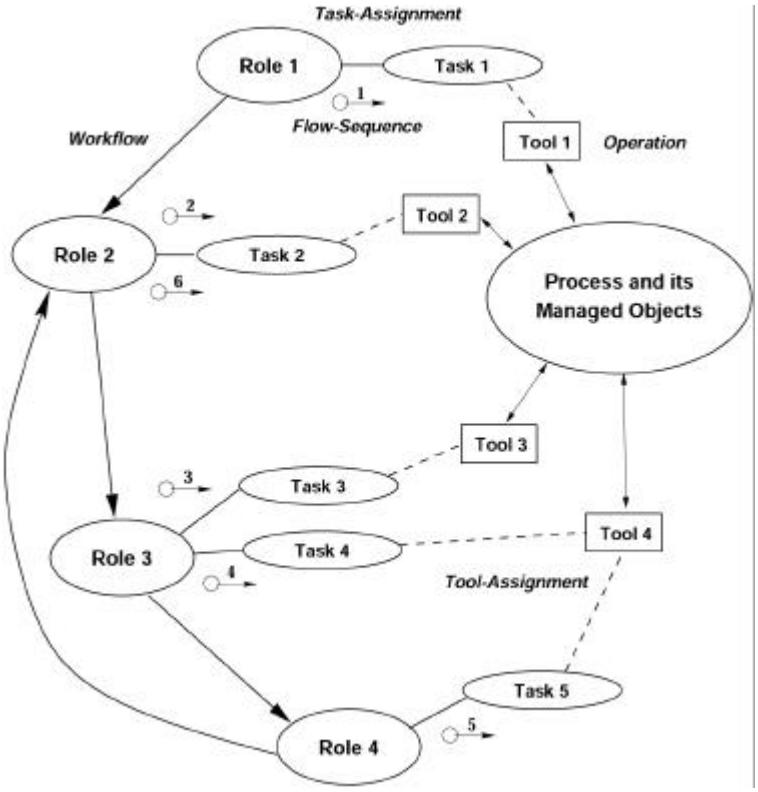

Figure 6: Workflow

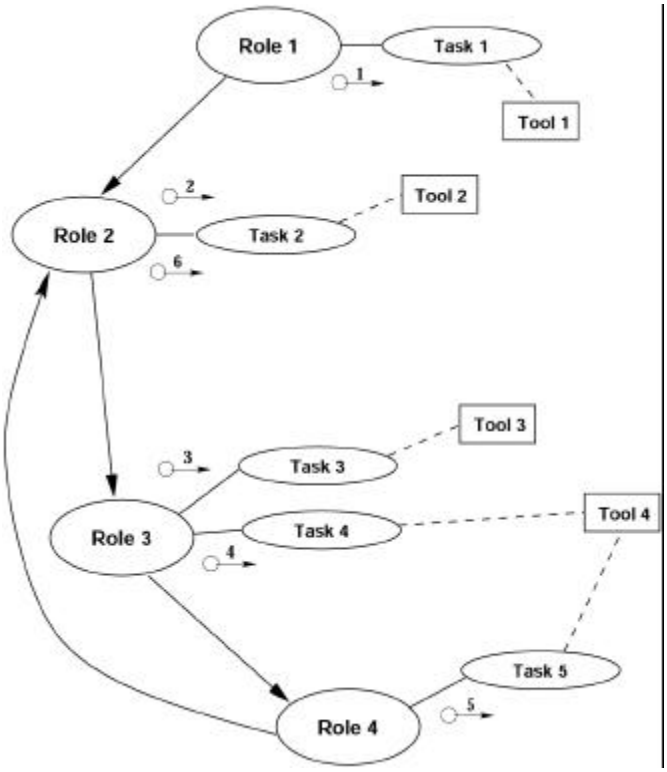

Figure 7: Example of a Workflow Scenario

7. Agents

An agent analysis describes or defines the characteristics of human operators (user analysis) or machine operators (functional system analysis) representing the roles of a management system. Human agents (users, operators) will be characterized by operator classes.

7.1 Operator Classes

The following characteristics representing one or more specific roles have to be defined:

Description: description of the task domains and qualifications of the operator class
Education: expected education for operators to be affiliated for the roles
Common-Knowledge: general knowledge background (general qualifications of operators)
Application-Knowledge: application knowledge needed for the execution of tasks (application qualifications of operators)
Skills: sensomotorical abilities in the handling of tools

It will be helpful to structure the operator classes into an hierarchy of classes, where members of the more general classes will be able to handle all tasks of the more specific classes which are derived from the classes they belong to (see Figure 8: Operator Classes and Operators). Management tasks and tools have to be designed and evaluated in accordance to the characteristics of the operator classes.

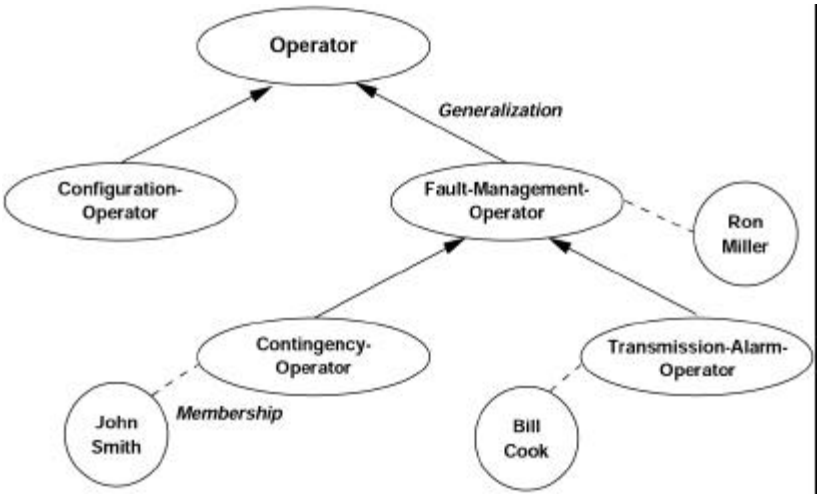

Figure 8: Operator Classes and Operators

7.2 Operator Class Assignment to Role Classes

For every role there has to be an associated agent class, in case of a human agent an operator class. The selected agent class has to provide abilities to cope with the tasks related to the role.

The task analysis will provide requirements (agent class attributes) for the operator classes that can be assigned for a role class.

8. Tools

Tools are support systems for the execution of tasks by operators. They should empower operators to perform the tasks in an efficient and correct way. In the area of interactive computer systems the tools will be whole application systems or dialogs within application systems.

Like in the case of tasks, roles, and agents, tools have to be structured into tool classes (e.g. *alarm-table*) and tool instances (e.g. *equipment-alarm-table*).

The definition, specification and implementation of specialized tools (tool classes) should be derived from the role classes, their associated task classes and the operator classes, i.e. tools should be defined quite late in the whole development process.

In many real situations tools will be there before the roles, tasks and operators are specified. If this is the case and there is no chance to change or replace the tools, the roles, tasks and operator qualifications will be determined heavily by these tools. This will lead usually to management systems of minor quality and should be avoided, since the cost of operating the system with these tools may be high and may even after a short period of operation even be higher than the cost for new tools.

The worst mistake which can be encountered commonly is, that new tools are designed before an analysis of tasks, roles and operators has been done, i.e. tools are designed without real application and organisation requirements.

Computer-based tools supporting the management of processes are sometimes called “management system” themselves, in the sense of systems to manage the process. Please note the more general meaning of “management system” incorporating at least application objects, tasks, roles, agents, and tools in this document.

In the area of telecommunication systems, management tools are structured in networks and are called the TMN (Telecommunication Management Network) [7]. In other application context they are sometimes called **decision support systems** or **process control systems** [12]. This framework has also be used for the analysis and design of tools in general business areas, with no emphasis on supervision and control of dynamic systems.

8.1 System Functions

Tools can be defined as sets of system functions each of them supporting the execution of a specific task by a user. Therefore the description of system functions can be viewed as a hierarchical decomposition of tools into sub-tools.

When we talk of system functions in respect to task analysis we always denote system functions from the operators point of view. These may not be necessarily equivalent to the system functions implemented in the system. For the operators tools provide specific perspectives or views to the system structure and the system functionality. Via the visible system functions the operators perform operations to solve the tasks to be done.

This kind of more functional analysis may take place after the task analysis has been done to a certain extent. The definition of functions is tightly coupled with the concepts of abstraction and decomposition of the system resources to be managed (see corresponding sections in this paper).

8.2 Tool Classes

The following characteristics have to be defined for each tool class:

Description: general description of the tool and its purpose

Tasks: list of tasks to be supported by tool

Input: input needed to apply tool

Output: output provided by tool

Sub-Tools: subordinate tools

Managed Resources: managed resources (managed objects) to be presented or manipulated

User Interface: description of the rules for the system presentation and the way how actions have to be performed:
this defines the look&feel of the interactive components of the management system
the user interface is defined by the following concepts Dialogs, Information Coding and I/O-Devices

Dialogs: syntactical rules of the user interface described as dialogs
(e.g. window handling, menu handling, data exchange, printing, user queries, customization dialogs, help dialogs, history dialogs)

Information Coding: description of presentation types and presentation characteristics
(e.g. naming, fonts, colors, icons, graphics, layouts, keybindings, formats)

I/O-Devices: description of the physical I/O-devices
the physical characteristics of keyboard, mouse, screen and other input/output devices

8.3 Tool Class Assignment to Role Classes

The assignment of tools to roles is predetermined by the provision of tools for the tasks the role has to perform. Therefore the tool assignment to roles will be done implicitly through the tool-task assignment.

Since usually many tasks will be performed by a role, the role's tools should not be just the sum of all tools that support these tasks. Instead of this, the tools should be optimized for the spectrum of all tasks to be performed. As a result of this optimization a **tool environment** has to be created, where all task oriented tool classes are integrated.

Through the tool assignment to roles it may show up, that a role will be confronted with a large number of tools or with rather complex ones. This may have impact to the operator classes assigned to these roles or may result in a reallocation of tasks between several roles.

9. References

[1] M. Herczeg, "Software-Ergonomie - Grundlagen der Mensch-Computer-Kommunikation". Addison-Wesley-Longman and Oldenbourg Verlag, Bonn, 1994.

[2] M. Herczeg, "A Task Analysis Framework for Mangement Systems and Decision Support Systems". In *Proceedings of the AoM/IAoM. 17th International Conference on Computer Science, Journal of Computer Science and Information Managment (CSIM)*, The International Association of Management (IAoM) and Maximilian Press Publisher, 1999, pp. 29-34.

[3] ITU, "Management Framework for Open Systems Interconnection (OSI) for CCITT Applications". *ITU-T Recommendation X.700*, International Telecommunication Union, Nov. 1992.

[4] ITU, "Common Management Information Service Definition for CCITT Applications". *ITU-T Recommendation X.710*, International Telecommunication Union, 1991.

[5] ITU, "Information Technology - Open Systems Interconnection - Structure of Management Information: Management Information Model". *ITU-T Recommendation X.720*, International Telecommunication Union, Jan. 1992.

[6] ITU, "Information Technology - Open Systems Interconnection - Systems Management: Object Management Function". *ITU-T Recommendation X.730*, International Telecommunication Union, Jan. 1992.

[7] ITU, "Overview of TMN Recommendations. *ITU-T Recommendation M.3000"*, International Telecommunication Union, Oct. 1994.

[8] B. Kirwan, L.K. Ainsworth, "A Guide to Task Analysis". Taylor & Francis, 1992.

[9] T.P. Moran, "Getting into a System: External-Internal Task Mapping Analysis". In *A. Janda (Editor), Proceedings of the CHI-83, Human Factors in Computing Systems*, Murray Hill, NJ, 1983, pp. 45-49.

[10] J. Rasmussen, "Strategies for State Identification and Diagnosis in Supervisory Control Tasks, and Design of Computer-Based Support Systems". In *Advances in Man-Machine Systems Research*, Vol. 1, JAI Press Inc., 1984, pp. 139-193.

[11] J. Rasmussen, "The Role of Hierarchical Knowledge Representation in Decisionmaking and System Management". *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-15(2), March/April 1985, pp. 234-243.

[12] J. Rasmussen, L.P. Goodstein, "Decision Support in Supervisory Control". *Technical Report M-2525*, Risø National Laboratory, Roskilde, Denmark, Aug. 1985.

[13] J. Rasmussen, L.P. Goodstein, "Information Technology and Work". In *M. Helander, editor, Handbook of Human-Computer Interaction*, Elsevier Science Publishers B.V. (North Holland), Amsterdam, 1988, pp. 175-201.

[14] J. Rasmussen, A.M. Pejtersen, L.P. Goodstein, "Cognitive Systems Engineering". John Wiley & Sons, New York, 1994.

[15] T.B. Sheridan, "Supervisory Control". In *G. Salvendy, editor, Handbook of Human Factors*, John Wiley & Sons, New York, 1987, pp. 1243-1268.

[16] T.B. Sheridan, "Task Allocation and Supervisory Control". In *M. Helander (editor), Handbook of Human-Computer Interaction*, Elsevier Science Publishers B.V. (North Holland), Amsterdam, 1988, pp. 159-173.

Michael Herczeg received his M.S. (Diploma) and Ph.D. degrees in computer science from the University of Stuttgart (Germany) in 1983 and 1986. From 1988 to 1996 he was a manager in software development departments in the telecommunication industry at Bosch Telecom and Alcatel.

Since 1996 he is professor for computer science and the director of the Institute for Multimedia and Interactive Systems at the University of Luebeck in Germany.

He is a member of IEEE, ACM, AAAI and the vice-chairman for Human-Computer-Interaction of the German Informatics Society (GI) and the German SIGCHI (ACM).