INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN
ICED 03 STOCKHOLM, AUGUST 19-21, 2003

CAPTURING PRODUCT REQUIREMENTS INTO THE TRANSFORMATION MODEL OF A TECHNICAL PROCESS BY USING UML

Tapio Korpela

Abstract

The design process of an original new product is started by an implementation free definition of the problem. The external properties of a product are commonly clarified at first. Typical users, user interfaces, and the typical using scenarios of the different kind of users are this kind of external properties. In my paper I am going to discuss and present ideas, how the third generation object-oriented graphical language (UML) can be applied for capturing product requirements into the formal models in early phases of the design process.

Key words: process modelling, requirements modelling, theory of technical systems

1 Introduction

In early phases of a design process the designer knows what the system should do, not how the system should be implemented or realised. The information processing science has developed object-oriented modelling methods for capturing the information and behaviour of entities in systematic models. The third generation object-oriented process modelling language: "Unified Modelling Language" (UML) [3] has been developed and adopted for the newest release of the model. The original designers of the Unified Modelling Language: James Rumbaugh, Ivar Jacobson and Grady Booch have developed earlier also their own object-oriented modelling methods, which are: the Booch method, published in 1991 [2], the Object Modeling Technique (OMT) by Rumbaugh et al., in 1991, and the Objectory (OOSE) by Jacobson et al., in 1992 [7]. The aim of the object oriented analysis method (OOA) is to make an implementation free model of the system. This model includes also the models of mechanical and electrical components. The same kind of basic idea of an implementation free model is seen also in the model of the technical process in the theory of technical system by Hubka and Eder [6].

2 Transformation system

A technical process is needed to transform something from an unsatisfactory state to a satisfactory state. In this process human beings, technical products and an active environment co-operate. An example of these transformation processes and products could be:

- *Mobile phone*: An urgent need to contact a distant person -> A person is reached for a conversation or a message is left for her or him, Vladimir Hubka and W. Ernst Eder, in their book: "Theory of Technical System" (TTS) [6], describes a total concept theory for engineering design. In this theory the product to be designed (technical system, TS, in Fig. 1) is one part of the transformation system, in which the technical process (Tp, in Fig. 1) is a formalised transformation process, which changes the state of operands (Od, in Fig. 1) in desired way. Transformation system consists of humans (Hu, in Fig. 1), environment (Env, in Fig. 1) and technical system, which co-operate with one another as well as with the technical process.

Technical process is decomposed into the partial processes, which are related to each other by the states of operands between consecutive processes. Transformations in these partial processes are realised by certain effects (Ef, in Fig. 1) , which are the means of achieving the desired transformation on operands. The effects are produced by the operators, which are: humans, technical systems and active environment. The active units, which create the necessary effects in the technical system are called organs and the organs are maintained in the final machine structure by parts. These three domains (transformation, organs and parts) are the basic elements of the Domain Theories (DT) developed by M. Myrup Andreasen [1].

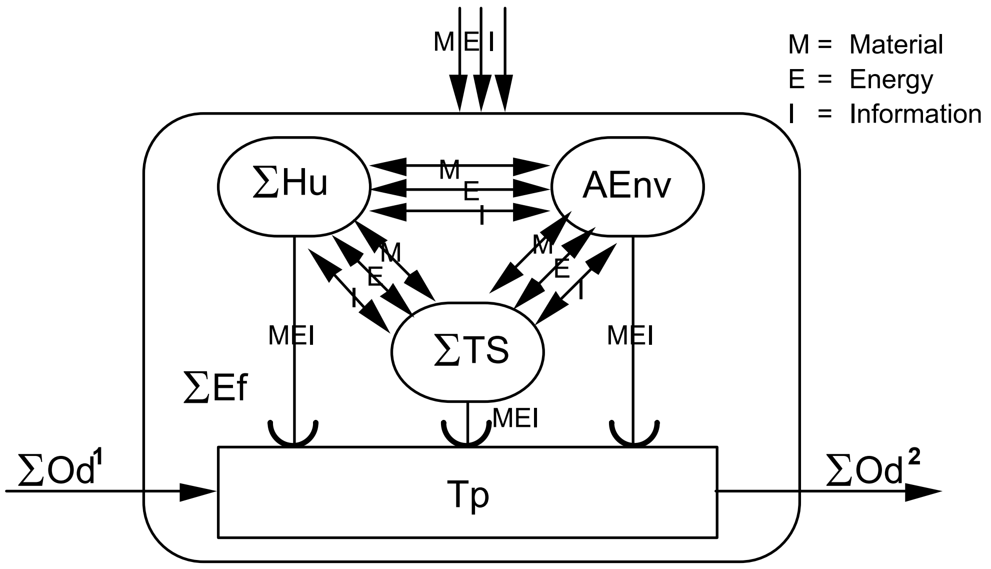

Figure 1. The transformation system by Hubka and Eder [6].

The most important part of this model is the state transition model of the Technical process, because the whole understanding of the problem to be solved is captured into it. Concluding from this matter, the crucial thing is to define the technical process. The first draft of the technical process for the problem can be a traditional preceding manual method, or a method, which has been used in the competitive product.

3 Object-oriented analysis

Object-oriented methods in software engineering are divided into implementation-independent analysis (OOA), implementation-dependent design (OOD) and implementation with a programming environment (OOP). Most important phase is analysis, because it is fully application-oriented and it is in this process that the logical customer-oriented models of the system are processed. The OOA process for software engineering, according to [7], is presented in Fig. 2.

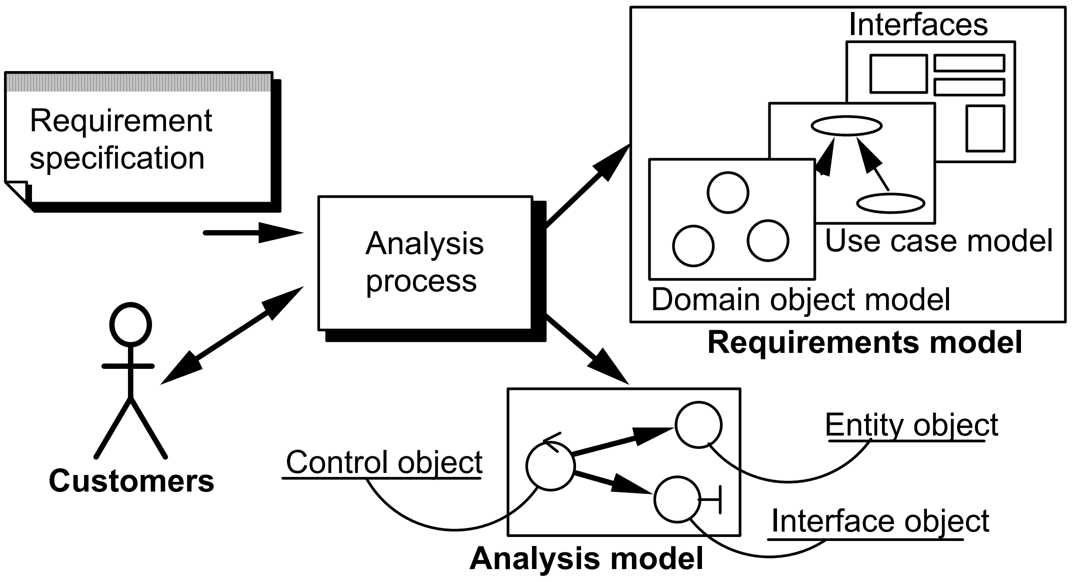

Figure 2. The OOA-process for system modelling by Ivar Jacobson [7].

The aim of object-oriented analysis is to model the system domain in a fully application-oriented manner. The implementation tools used, e.g. DBMS and programming source codes, do not constrain the OOA process. Analysis starts with requirements modelling. In requirements modelling of a system, the external properties of a product are commonly clarified at first. Typical users, user interfaces, and the typical using scenarios of the different kind of users are these kind of external properties. OOA offers formal methods for this definition process in carrying out using actors, use cases, classes, and sequence and collaboration diagrams, which is called the Use Case development [4]. The requirements of a design system are clear and easy to refine to object models with end users, because the semantic gap between the model and reality is small [7]. The notations used in the modelling process are usually simple and easy to understand. In Fig. 3 there has been presented the principle idea of the object oriented analysis process.

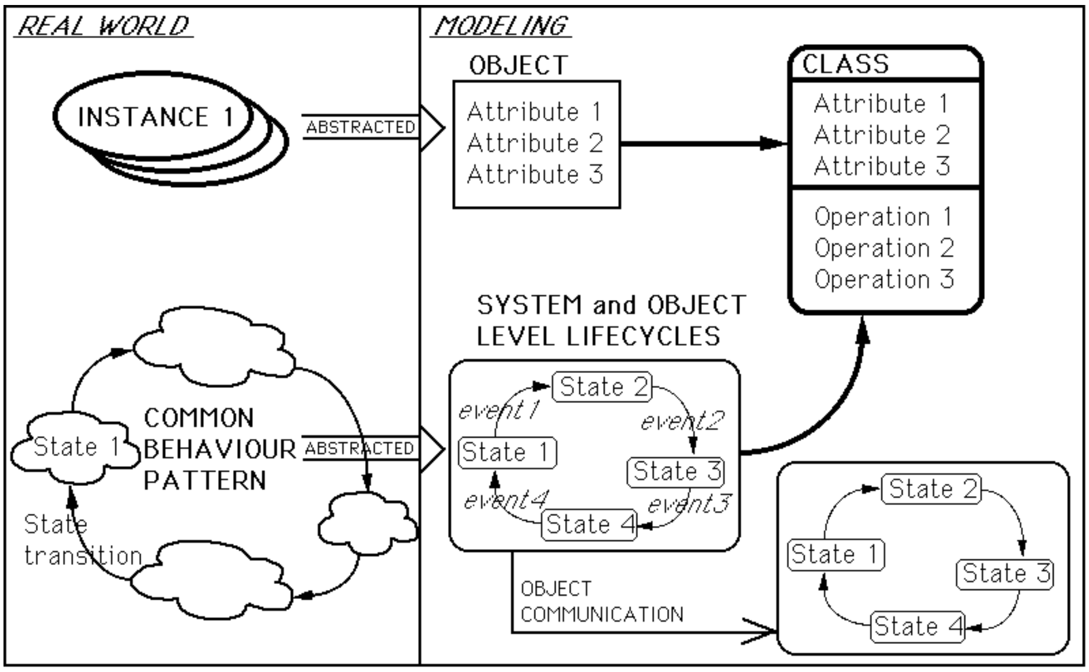

Figure 3. From real world entities to the models of objects and classes.

The real world entities, called instances in Fig. 3, are modelled by the list of attributes. These instances have their common behaviour patterns, which can be modelled by state transition diagrams. The attribute structure and common behaviour of similar instances are defined in their common classes [2]. In the transformation model by Hubka and Eder (presented in Fig. 1.), the needed effect f(Ef) to achieve the transformation from initial unwanted state Od^1 to the final desired state Od^2 is produced by the technical system, human or the active environment. Similar to this principle is to abstract the common behaviour pattern to produce the lifecycle of an unspecified instance of a system or an object by using state transitions diagrams in OOA.

The UML notation encompasses [3]:

- *things*, which are structural things such as classes, interfaces, collaborations, use cases, active classes, components, and nodes; behavioural things such as interactions and state machines; grouping things which are packages; and annotational things which are notes.
- *relationships*, types of which are: dependency, association, and generalization.
- *diagrams*, which are: class, object, use case, sequence, collaboration, statechart, activity, component, and deployment diagram.
- *extensibility*, which consists of three mechanisms for extending the UML language syntax and semantics: stereotypes, tagged values, and constraints.

4 The chain from state transition to the structure

According to German Gerhard Pahl and Wolfgang Beitz [8] a system has boundary, which connects it to its environment by means of inputs and outputs. A system can be divided into sub-systems. It is possible to define appropriate system at every stage of abstraction, analysis or classification. As a rule such systems are parts of larger, superior systems. All technical systems involve the conversion of energy, material and/or signals, which must be defined in quantitative, qualitative and economic terms [8].

First the system is described by an overall function, which is abstracted from the technical specification. The overall function is divided into the sub-functions. This process leads to the functions structure (*functional interrelationship*). Sub-functions are solved by means of *working interrelationships*. These are effects of physical (chemical, biological) phenomena or geometric or material characteristics to fulfil the sub-functions. The result is the working structure, which is called the organ structure in Theory of Technical Systems (TTS) by Hubka and Eder [6] and in Domain Theories (DT) by Andreasen [1].

Parts and assemblies, which realize and maintain the working structure form *a constructional interrelationship*. The construction to be designed usually belongs to a superior system. It has *a system interrelationship*. In this level the product has its contact to the users and active environment. The superior systems effects also internally to the lower level constructions.

If an electric shaver is taken as an example for the product definition, the primary problem is: “How the beard on the face is cut as short as possible without damaging the skin under the beard?” As an active environment the electric power is available from the plug or from included loadable batteries. The body of the device can be manufactured handy and light. The electric power is able to transform easily to reciprocating or rotating mechanical movement.

The critical construction is the cutter head. When starting the definition, the scenarios of the problem: “How to shave?” can be written. The designer has images of practical user interfaces for the device. These matters outline the first draft of the implementation free model of the technical process. The technical process is a state transition model, which is accomplished during the design process. The development of the needed technical system is the conclusion chain from needed effects to make state transitions happen in the technical process to the means, which are realized by the machine constructions and other technical solutions, as it is presented in the next Fig. 4.

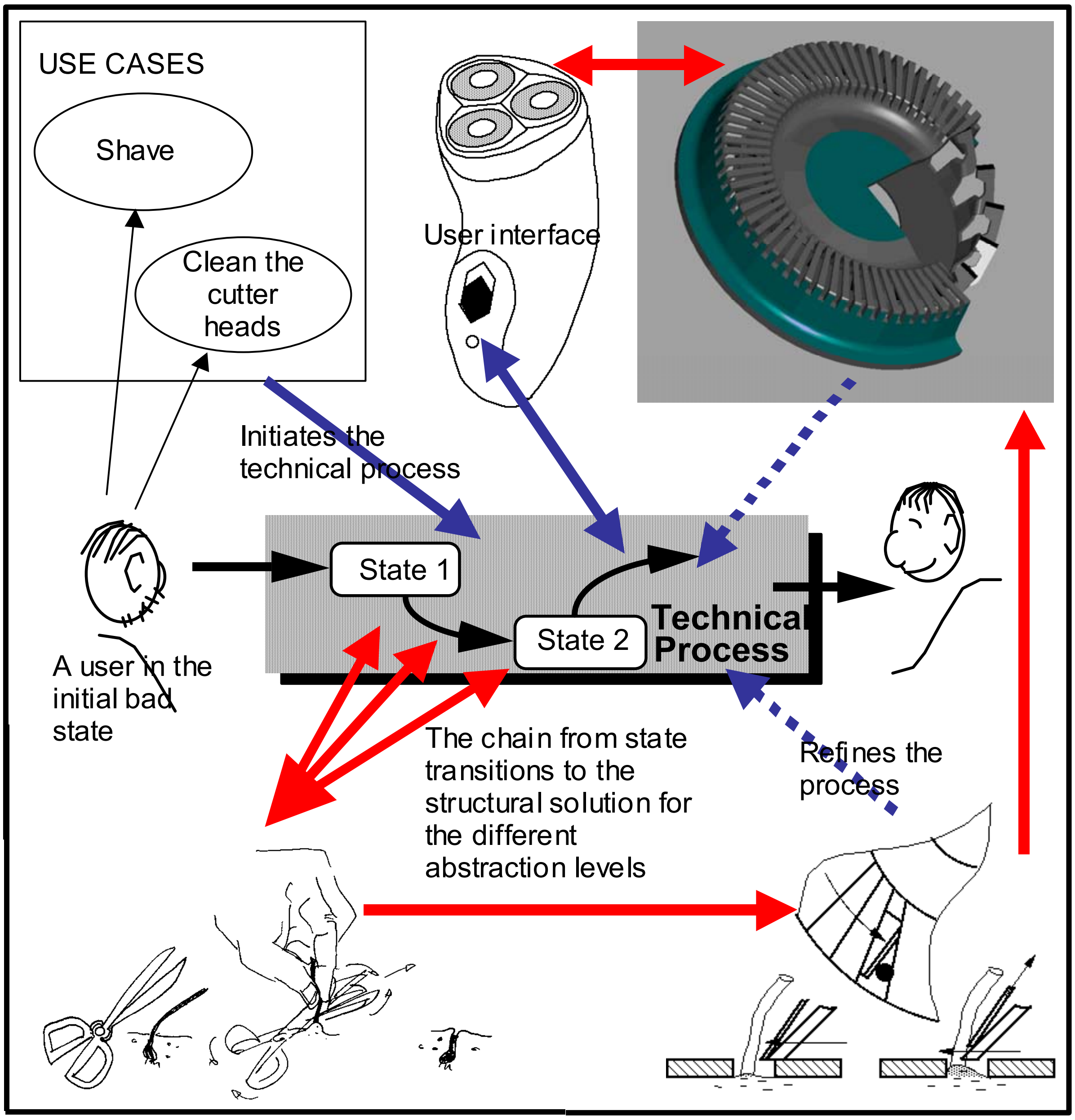
USE CASES
Shave
Clean the cutter heads
User interface
Initiates the technical process
A user in the initial bad state
State 1
State 2
Technical Process
The chain from state transitions to the structural solution for the different abstraction levels
Refines the process

Figure 4. Design data is maintained and managed in the state transition model of the technical process.

Modern products include very often besides mechanics also electrical components and embedded software. The automation (hardware and software) gets stimuli from the environment by the help of sensors such as thermocouples, optical scanners and contact probes, for example. An effector is the part of mechanical, electrical, optical equipment or components to make an effect that causes some chances in the environment into which these components belong to.

The design process of an original new product is started by the implementation free definition of the problem. Use Case development, which is used in Unified Modeling Language (UML) and presented also in the previous pages of this paper, offers powerful formal tools for modelling the requirement of the product. The results of the definition are captured into the state transition model of the technical process, as it is presented in the next Fig. 5. The means to carry out the state transitions in the technical process are realized by software, or electrical hardware, or mechanical constructions.

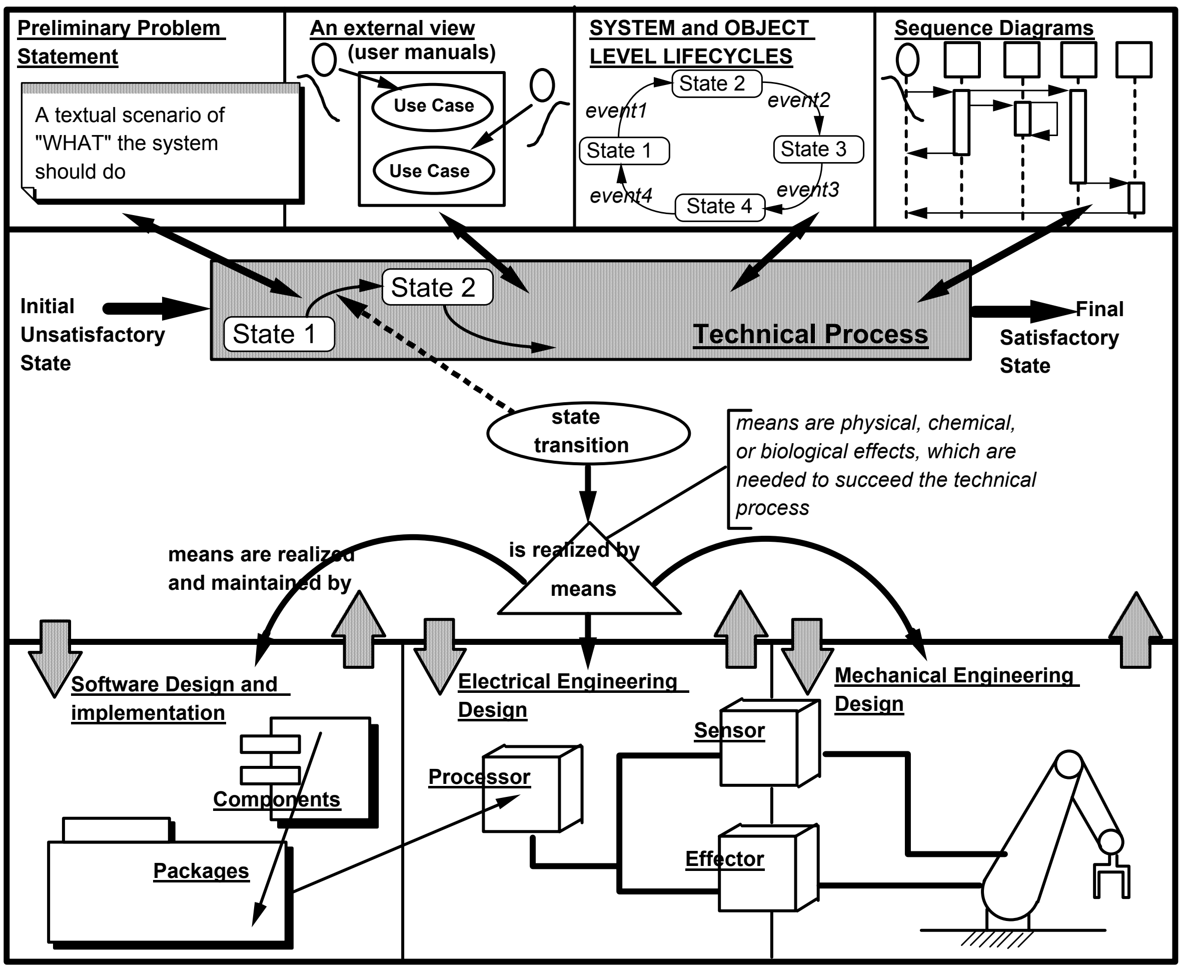
Preliminary Problem Statement
A textual scenario of "WHAT" the system should do
An external view (user manuals)
Use Case
Use Case
SYSTEM and OBJECT LEVEL LIFECYCLES
State 1
State 2
State 3
State 4
event1
event2
event3
event4
Sequence Diagrams
Initial Unsatisfactory State
State 1
State 2
Technical Process
Final Satisfactory State
state transition
means are physical, chemical, or biological effects, which are needed to succeed the technical process
is realized by
means
means are realized and maintained by
Software Design and implementation
Components
Packages
Electrical Engineering Design
Processor
Sensor
Effector
Mechanical Engineering Design

Figure 5. The layered design process of a mechatronic product.

Design tools for the software, the electrical engineering, and the mechanical engineering includes their own class hierarchies and libraries for modelling and implementing the detail structures in these different disciplines of the technology. The unifying layers of the product modelling presented in Fig. 5 are the state transition model of the technical process and the models of the use case development, which are accomplished during the design process.

5 An example

The problem is to define and develop a self-service cocoa maker maintained by the bartender and used by an ordinary client as a self-service machine in a cafe. The first draft of the technical process can be read from a commercial package of cocoa powder under the heading "*Directions for use*".

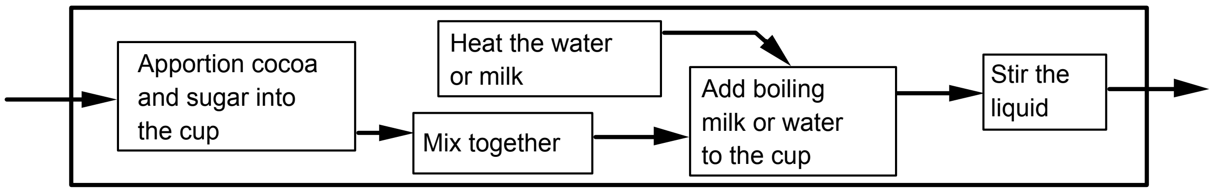
Apportion cocoa and sugar into the cup
Heat the water or milk
Mix together
Add boiling milk or water to the cup
Stir the liquid

Figure 6. The first draft of the technical process for making a cup of cocoa.

Some obvious and tangible objects, which have direct counterparts in the modelled world (Domain objects), can be perceived. Water tank, heater, boxes for cocoa, sugar and dry milk, apportion systems and mixing system are needed for example in the cocoa maker. More objects can be found by using a top-down approach to requirement modelling, whereupon new objects appear throughout the OOA process.

The UML offers for the requirements modelling: the use case models, sequence diagrams, and statecharts of the use cases. In this case there are two typical users (actors) and their use cases, namely the bar client, who makes the cup of cocoa with the system and the bartender, who keeps the cocoa maker up. Simple user interfaces for the client and the bartender can be described.

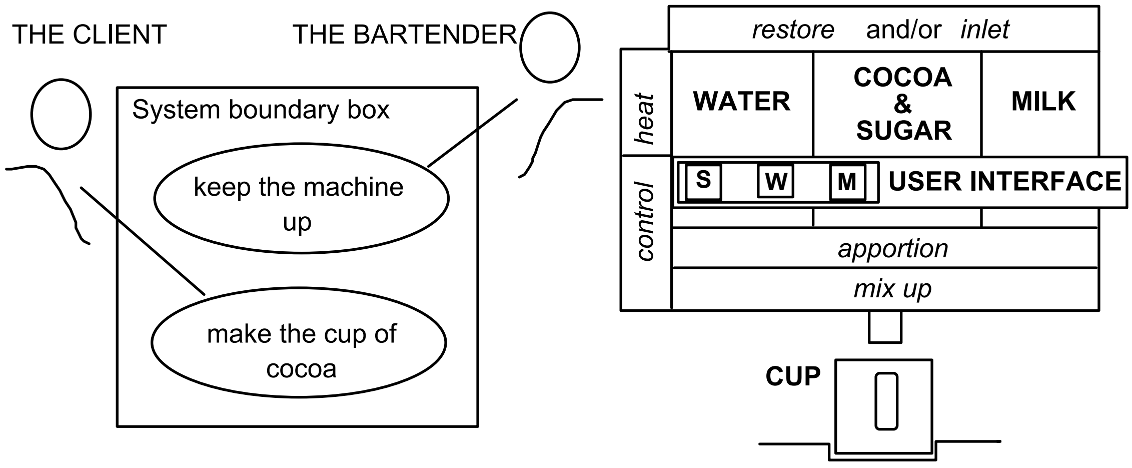

Figure 7. The actors, their use cases and the preliminary layout of the cocoa maker.

The more interesting and meaningful is the use case of the client. In Fig. 7 there has been presented the first draft of the simple user interface of the client. The interface consists of the start button marked with a letter “S”. There are also the selection buttons for milk (marked with “M”) and for water (marked with “W”). The textual description for the client’s use case, which is also the directions for use, could consists of the following lines:

1. Put an empty cup under the tube.
2. Press the start button marked with “S” in the panel.
3. Wait until the milk and water buttons marked with “M” and “W” start to blink in the panel.
4. Select the liquid into your cocoa by pressing milk or water button in the panel.
5. Wait until the lights in the buttons are gone out.
6. Take the cup out of the machine table.

The use cases and their textual description are external views of the system. The technical process can be refined by modelling the behaviour of the system. Statecharts are formal graphical language for defining the behaviour of the use case. In modelling the statechart of the clients use case (Fig. 8), there has been applied Sally Shlaer's and Stephen J. Mellor's state transition modelling method [9].

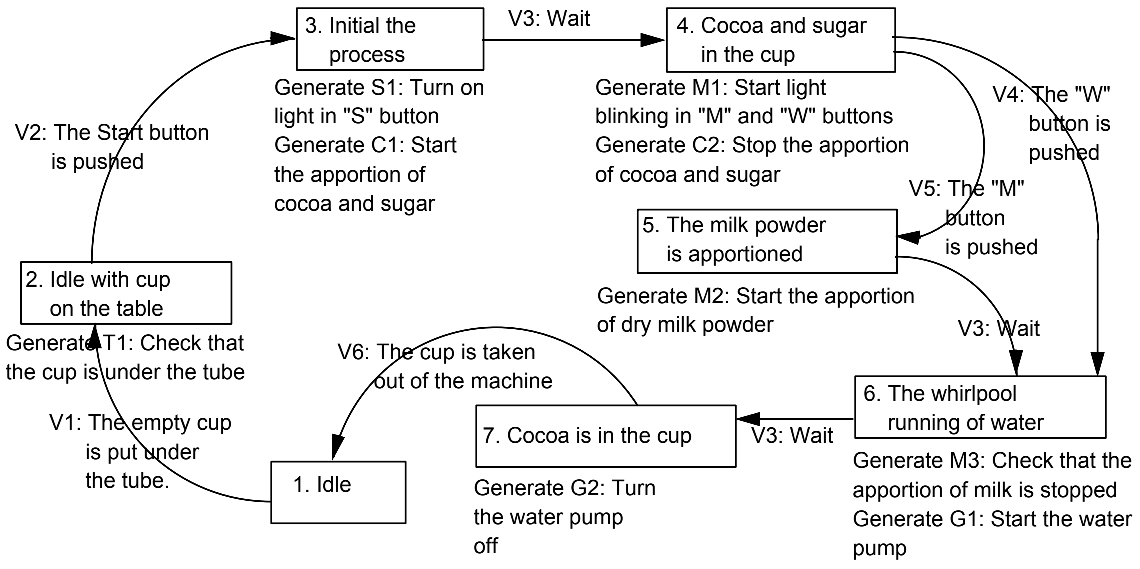
3. Initial the process
Generate S1: Turn on light in "S" button
Generate C1: Start the apportion of cocoa and sugar
V3: Wait
4. Cocoa and sugar in the cup
Generate M1: Start light blinking in "M" and "W" buttons
Generate C2: Stop the apportion of cocoa and sugar
V4: The "W" button is pushed
V2: The Start button is pushed
V5: The "M" button is pushed
5. The milk powder is apportioned
Generate M2: Start the apportion of dry milk powder
2. Idle with cup on the table
Generate T1: Check that the cup is under the tube
V3: Wait
V6: The cup is taken out of the machine
6. The whirlpool running of water
Generate M3: Check that the apportion of milk is stopped
Generate G1: Start the water pump
V1: The empty cup is put under the tube.
7. Cocoa is in the cup
Generate G2: Turn the water pump off
V3: Wait
1. Idle

Figure 8. State model of the cocoa making process.

A state transition diagram (STD) for the cocoa maker is presented in Fig. 8. The states are represented in rectangular boxes. The states have been written in these boxes and they are labelled with order numbers. The transitions are shown by arcs, which connect the two states. Each transition is labelled with the letter "V", order number and the event that causes the transition. The texts under the state boxes labelled with the text "Generate" are the actions, which are associated with the state above. An action is an activity or operation that must be accomplished when an instance (object) arrives in a state. State modelling method used in this example differs from the method used in UML. The state model of Shlaer & Mellor [9] bases on Moore form, which allows actions only upon state entry. The UML notation includes also actions when a transition is taken (Mealy form) and activities, which takes place during the states. States can be nested and the transitions can be modelled more precisely in the UML [5].

In this example the statecharts has been applied for modelling the behaviour of the use cases. Later in OOA process the statecharts are applied to model the behaviour pattern of one object. Actions associated with the states gives the functions of the objects. Besides this objects have static relationships and they communicate with each other. Typical static relationship among objects can be aggregation (part of) or inheritance (kind of). The water tank and the heater element are parts of the cocoa maker. Start button is a kind of push button. Events between objects and external entities such as operators (users of the system), physical devices, and objects in other subsystems are modelled by the object collaboration models.

The sequence diagram in the UML is used to model the flow of control in the system. It shows the sequence of messages between the objects in the system. There are two scenarios for the client's use case. The cup of cocoa can be made with or without milk. In the next Fig. 9 the case with milk has been presented.

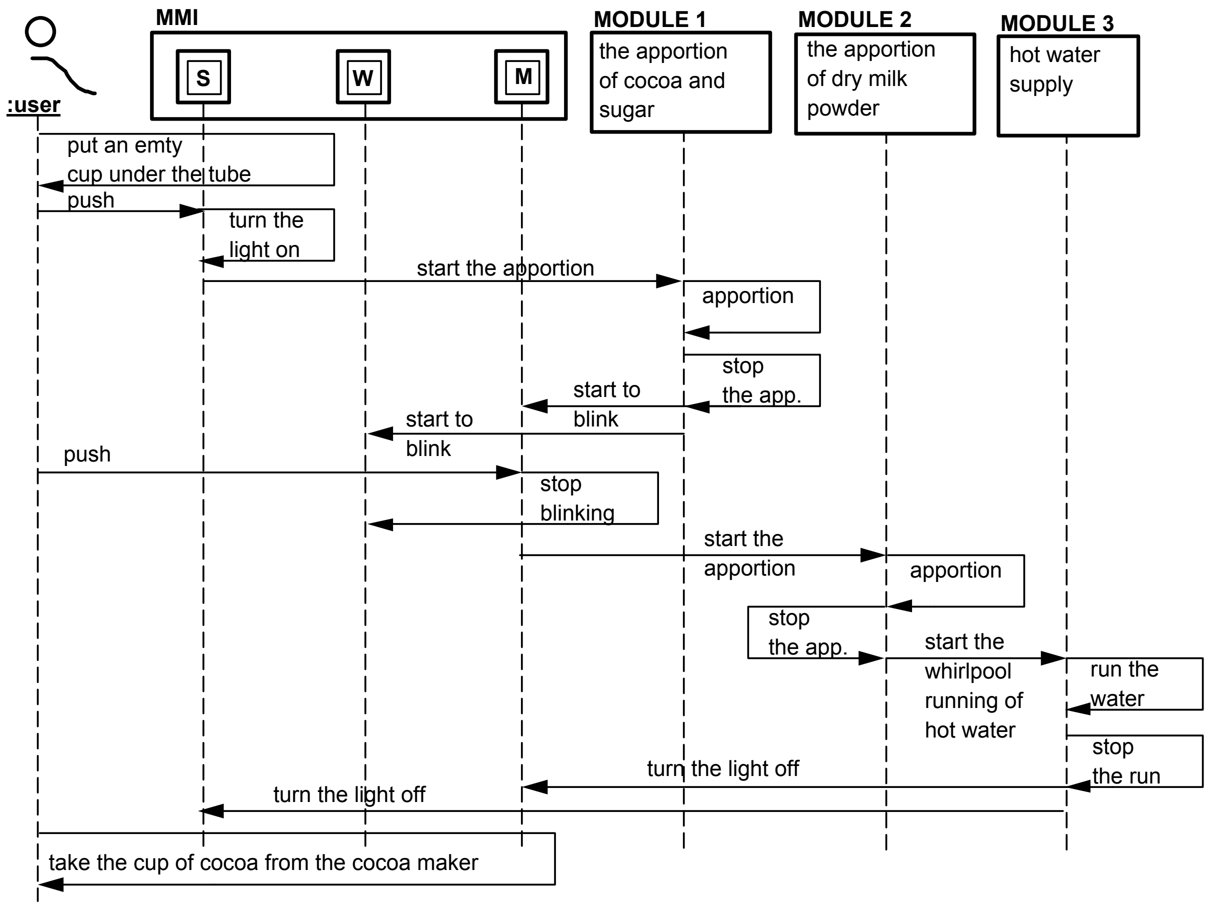

Figure 9. The sequence diagram for making a cup of cocoa with milk.

Co-operative actors, user interface objects and the known domain objects are named in the upper side of the diagram and drawn as vertical lines (dashed lines in Fig. 9). Horizontal arrows with texts are the messages. The sequence diagram allocates the needed functionality of the product (operations) to the participating objects.

6 Conclusion

The design process of an original new product is started by the implementation free definition of the problem. The external properties of a product are commonly clarified at first. Typical users, user interfaces, and the typical using scenarios of the different kind of users are these kind of external properties. The third generation graphical OO modelling language UML offers formal methods for this definition process in carrying out using actors, use cases, classes, and sequence and collaboration diagrams, which is called the Use Case development [4]. The results of the definition are captured into the transformation (state transition) model of the technical process. The means to carry out the state transitions in the technical process are realised by software, or electrical hardware, or mechanical constructions. Design tools for the software design, the electrical engineering design, and the mechanical engineering design include their own class hierarchies and libraries for modelling and implementing the detail structures. The integrating layers of the design process are the state transition model of the technical process and the models of the use case development, which are accomplished during the process. In this way the models for defining the product corresponds always with the designed construction during the process, and the chain from technical requirements to constructions can be traced from the models.

Modelling tools, which base on UML, have been developed for modelling the software for information systems and embedded control for mechatronic products. In these tools software and electrical modules of products are modelled and managed well from requirements into the implementations. Mechanical modules have been considered only by their I/O interfaces in these system models. Characteristics of mechanical solutions have to be taken into account from the first line of product development to the final structural solutions so that the best combination of the solution principles are found out in these modelling approaches.

References

[1] Andreasen, M., M.: "Design Methodology", a PhD course on theory, research and education, Technical University of Denmark, Denmark, 1998.

[2] Booch, G.: "Object-Oriented Design with Applications", The Benjamin/Cummings Publishing Company Inc., California, 1991.

[3] Booch, G., Rumbaugh, J., Jacobson, I.: "The Unified Modeling Language User Guide", Addison Wesley Longman, Inc., USA, 1999, 482 p.

[4] Canals, A.: "Use of UML/CS SI Development Process", JOOP, 13(12): 10-17, April 2001.

[5] Douglas, B., P.: "Real-time UML: developing efficient objects for embedded systems", Addison-Wesley Longman Inc., United States of America, 2000.

[6] Hubka, V. & Eder W.,E.: "Theory of Technical Systems", Springer-Verlag, New York, 1988.

[7] Jacobson, I., Christerson, M., Jonsson, P. & Overgaard, G.: "Object-Oriented Software Engineering: A Use Case Driven Approach", Addison-Wesley, Wokingham, England, 1992.

[8] Pahl, G., Beitz, W.: "Konstruktionslehre: Handbuch für Studium und Praxis" 2. Auflage, Springer-Verlag, Berlin, 1986, 590 p.

[9] Shlaer, S. & Mellor, S.,J.: "Object Lifecycles: Modeling the World in States", Englewood Cliffs, New Jersey, Yourdon Press, 1992.

Corresponding author:
Tapio Korpela
University of Oulu
Department of Mechanical Engineering
P.O.Box 4200, FIN - 90014 University of Oulu, Finland
Tel: +358 8 553 2052, Fax: +358 8 553 2026, E-mail: Tapio.Korpela@me.oulu.fi