Tanja Mehtonen

IMPROVEMENT OF COURSE ENROLMENT PROCESS

Case: Oulu University of Applied Sciences, School of Business and Information Management

IMPROVEMENT OF COURSE ENROLMENT PROCESS

Case: Oulu University of Applied Sciences, School of Business and Information Management

Tanja Mehtonen
Bachelor's thesis
Autumn 2010
Business Information Technology
Oulu University of Applied Sciences

ABSTRACT

Oulu University of Applied Sciences
Degree Program in Business Information Technology

Author: Tanja Mehtonen
Title of thesis: Improvement of course enrolment process Case: Oulu University of Applied Sciences, School of Business and Information Management
Supervisor: Liisa Auer
Term and year when the thesis was submitted: Autumn 2010
Main text + appendices: 56 + 4

ABSTRACT

From a process-related perspective, an organization can be seen to be an entity where the focus is on the organizations tasks or actions. In order for the organization to fulfil its goals the processes within need to be as simplified as possible in order to make them effective. Process management is used to control and improve processes.

The aim of the bachelor's thesis was to analyse and model the current process of course enrolment of Oulu University of Applied Sciences in order to discover the underlying problems of it. The purpose was to introduce a solution to these problems by designing and developing a database table to improve the management of course information together with a new user interface for the students course enrolment. The information management department of Oulu University of Applied Sciences was the commissioner of the thesis work.

The waterfall software development process model was used as the basis for the development task of the thesis. UML was used in order to model the current process. PL/SQL was the main programming language used together with the software Toad for Oracle 9.1. The thesis author had experience of using the software and PL/SQL during practical training done for the commissioner.

As a result of this thesis, the commissioner received a working student user interface for course enrolment and a database table for the management of course information. The new students user interface is ready to be taken into test use at the School of Business and Information Management of Oulu University of Applied Sciences.

Key words: Asio, eHOPS, course enrolment, process, process management, database, PL/SQL

TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu
Degree Program in Business Information Technology

Tekijä: Tanja Mehtonen
Opinnäytetyön nimi: Improvement of course enrolment process Case: Oulu University of Applied Sciences, School of Business and Information Management
Työn ohjaaja: Liisa Auer
Työn valmistumislukukausi ja -vuosi: Syksy 2010
Sivumäärä: 56 + 4

TIIVISTELMÄ

Prosessinäkökulmasta organisaation voidaan katsoa olevan kokonaisuus, jossa pääpaino on organisaation sisäisissä tehtävissä ja toiminnoissa. Organisaation sisäisten prosessien täytyy olla mahdollisimman pelkistettyjä, jotta ne toimisivat tehokkaasti ja organisaatio täyttäisi tavoitteensa. Prosessin hallintaa käytetään prosessien ohjaukseen ja parantamiseen.

Opinnäytetyön tavoitteena oli analysoida ja mallintaa Oulun seudun ammattikorkeakoulun nykyinen opintojaksoille ilmoittautumisprosessi siinä esiintyvien ongelmien selvittämiseksi. Tarkoituksena oli esitellä ratkaisu löydettyihin ongelmiin suunnittelemalla tietokantataulu kurssien hallinnoinnin parantamiseen, sekä uusi käyttöliittymä opiskelijoiden opintojaksoille ilmoittautumiseen. Oulun seudun ammattikorkeakoulun tietohallinto oli opinnäytetyön toimeksiantaja.

Opinnäytetyön kehittämistyön pohjana käytettiin vaiheellista ohjelmistokehityksen vesiputousmallia. Nykyinen prosessi mallinnettiin käyttämällä UML-mallintamistekniikkaa. Toteutusvaiheessa käytettiin PL/SQL-ohjelmointikieltä yhdessä Toad for Oracle 9.1. työkalun kanssa. Opinnäytetyön tekijällä oli aiempi tuntemus käytetystä ohjelmointikielestä ja ohjelmistosta toimeksiantajalla suorittamansa työharjoittelun kautta.

Opinnäytetyön tuloksena toimeksiantaja sai toimivan käyttöliittymän opiskelijoiden opintojaksoilmoittautumiseen, sekä tietokantataulun kurssien hallinnointiin. Uusi opiskelijoiden käyttöliittymä on valmis otettavaksi testikäyttöön Oulun seudun ammattikorkeakoulun liiketalouden yksikössä.

Asiasanat: Asio, eHOPS, opintojaksoilmoittautuminen, prosessi, prosessin hallinta, tietokanta, PL/SQL

CONTENTS

1 INTRODUCTION

The course enrolment for the common free choice studies and the free choice studies of Oulu University of Applied Sciences is done via Asio EduERP-student management software developed by a Finnish company Asio-Data Oy for the Universities of Applied Sciences. The software is based on an Oracle database and Oracle programming languages and is almost completely Web-based.

The main parts of the software are schools operations planning and control, which includes course plan preparations. Student logistics part which includes the student's individual services such as a personal study plan, presence/absent registration, course and exam enrolments, study register, feedbacks and diploma applications. Study office functions which are used for common choices, dynamic study cards and diplomas, reporting, international mobility, training and feedback handling. Evaluation part includes the monitoring of partially completed courses, grades for a study period and studies. The software is in use in Jyväskylä, Mikkeli, Kajaani and Oulu Universities of Applied Sciences. (Asio-opiskelijahallinto-ohjelmiston sisältö. Asio, 2010, date of retrieval 10.2.2010.)

The software has several problems which appear in the course enrolment process of Oulu University of Applied Sciences and are mainly related to the student logistics part. This fact generated an interest in the commissioner of the thesis work, the information management department of Oulu University of Applied Sciences to improve the current process of course enrolment in order to make the process more dynamic and efficient, and to better meet the requirements of the university.

There are many unnecessary delays in the current course enrolment process and a lot of futile work is done in order to make the enrolment for the courses possible for the students. At the moment any changes which need to be done to the course information need to be coded directly into the source code of Asio EduERP making the management of the system very difficult. There are currently several different date notations used for the availability dates set for the courses which makes the changing of the dates difficult as it is hard to remember which date notation to use in a specific source code package. The current system is also very static as the course offerings which can be done with the system are very limited. Also any modifications which need to be made to the course offering are not easy to be made and tailoring of the courses offered via Asio EduERP is difficult. (De Bruijn, Koivukoski & Malinen 14.1.2010, discussion.)

The students of Oulu University of Applied Sciences create a personal study plan, HOPS during their first year of study where the progress of their studies can be planned. EHOPS, electronic personal study plan is a tool used in creating the personal study plan. With the help of eHOPS students can easily follow the progress of their studies and create a frame for their studies by choosing courses offered to their degree program into their study plan. Currently the courses selected into the personal study plan, eHOPS do not appear anywhere during the enrolment process and it was seen by the commissioner as helpful for the students to be able to see the eHOPS course selections when enrolling for the common free choice studies and free choice studies. This would help the student in their course selections and study planning. (What is eHOPS. Oulu University of Applied Sciences 2010, date of retrieval 24.2.2010.)

The aim of the thesis work was to improve the course enrolment process of Oulu University of Applied Sciences. Therefore the main goal was to analyse and model the current process of course enrolment in order to discover the underlying problems of it and to introduce a solution to the problems. A solution to some of the problems of the current process was introduced in the form of a new database table which would improve the management of course information. Not only does this new database table give a new method for the staff to manage and administrate the course enrolment process, it also introduces a new user interface for the students to enrol to all offered courses by using one user interface. This user interface replaces the two existing user interfaces for the enrolment for Oulu University of Applied Sciences; common free choice studies and the free choice studies of the school, and also lists the students selected eHOPS courses. (De Bruijn et al. 14.1.2010, discussion.)

An Oracle based database, which gathers information of the courses used in the course enrolment process, was developed. The database table enables the staff to add, remove and modify course availability and course information of those courses which will be set for enrolment in the new student user interface. The main focus of the development phase was to design the database table and to create a new user interface for the students. The use of the new database table makes the setting of availability dates and adding information for the common free choice study courses and the free choice study courses of Oulu University of Applied Sciences more dynamic and administrable.

When narrowing down the scope of the thesis it was decided that the user interface for the database table would not be included in the thesis work. Therefore the developed database table is accessed by using the Oracle-software currently utilised in the information management department for the administration, modification and development of the databases used in the Oulu University of Applied Sciences. However the user interface for the database table should be developed in the future in order to fully realize the improvements which it brings to the course enrolment process. (De Bruijn et al. 14.1.2010, discussion.)

In the beginning of the thesis work a decision was made in which it was stated that in the future the students will need to register to all of the courses offered by Oulu University of Applied Sciences. These courses are also listed in the new student user interface and this needed to be taken into account when specifying the requirements for the software. Both the database table and the students user interface have an interface with eHOPS, the electronic study plan, the course register and Asio EduERP, and therefore utilise the information of these existing systems. The database table and the user interface for course registration are designed for the use of School of Business and Information Management of Oulu University of Applied Sciences which will be acting as the case study in the thesis work. (De Bruijn et al. 14.1.2010, discussion; Hedemäki 17.3.2010, discussion.)

The theoretical foundation of the thesis consists of description of different process types, basic steps of process management, process modelling and different modelling notations used in process modelling. Theory is then put into practice in the modelling and analysis of the current process of course enrolment of Oulu University of Applied Sciences in order to discover the underlying problems of the process. A solution to these problems is then introduced in the form of a database table for course enrolment process management and a new user interface for the students. The database table and the user interface together with their parts in the current process are modelled, analysed and further explained. The development phase of this thesis work is conducted according to the waterfall software development model which gives a systematic sequential approach to software development. In the waterfall model the software development process goes through the following sequential flow of phases; requirements, analysis, design, implementation, integration and deployment and maintenance (Pressman 2005, 79-80).

2 RAKETTI PROJECT

Oulu University of Applied Sciences is currently taking part in a project called Raketti which is a joint project between the Finnish higher education institutions and the Finnish Ministry of Education with the aim of improving the quality, compatibility, and usability of information and IT solutions in the steering and monitoring of higher education and in the management of higher education institutions. The focus of the project is on the enterprise architecture of the higher education institutions. The project consists of four subprojects: KOKOA with the focus on overall architecture model for the higher Education System, OPI where the focus is on the study administration information system, XDW where the focus is on concept definition, concept model and data warehouse, and TUTKI with the focus on research administration. (Raketti, rakenteellisen kehittämisen tukena tietohallinto 2010, date of retrieval 31.3.2010.)

As the current process of course enrolment in this thesis work will be described from the operational process point of view in order to improve the administration of the data, the OPI subproject is seen as relevant to this thesis work. The OPI subproject seeks to increase collaboration among higher education institutions in study administration and related information systems. The project uses the overall architecture of study administration as the framework in examining the study administration system from the perspective of operational processes and related information as well as information systems and related technologies. After this, based on the selected operating model, the project goes on to identify the future needs and challenges for study administration in higher education and the common processes which can be supported with information management. (Opi-tavoitemuistio_1.0 2010, date of retrieval 31.3.2010; Koivukoski 14.1.2010, discussion.)

The main objectives of the OPI subproject are to firstly define and create a common basic system for study administration and a reference architecture in case of outsourcing for the higher education institutions. Secondly to develop the information system in study administration by adopting a national concept and informational model to ensure consistent and compatible information. Thirdly the OPI subproject seeks to enhance the national coordination of development projects in study administration information systems at higher education institutions and finally to make the subsystem created during the project available for free through a national software pool. (Opi-tavoitemuistio_1.0 2010, date of retrieval 31.3.2010.)

If successful in fulfilling the objectives the OPI subproject will enhance the knowledge-based management and aid the coherent reporting to authorities of higher education institutions. It will also enable the higher education institutions to focus on their core operations. In the long run the objectives will improve the compatibility of study administration information systems, increase cost-efficiency in system maintenance and development, and reduce overlapping work. The higher education institutions involved in the project agree upon commonly used codes, information structures and data entry practices which will in turn enable comparable, up-to-date information in support of local management and national steering. (Opi-tavoitemuistio_1.0 2010, date of retrieval 31.3.2010.)

The course enrolment process of Oulu University of Applied Sciences has not been modelled before and no current description of the process exists. Therefore the modelling and analysis of the current process of course enrolment done in this thesis work is important and can be used as a part of the Raketti project, OPI subproject when modelling and analysing different study administration processes of Oulu University of Applied Sciences. If succesfull the thesis work could also be used to upgrade the systems of other universities which currently use Asio EduERP student management software in their course enrolment processes. (Koivukoski 14.1.2010, discussion.)

3 PROCESS AND PROCESS MODELLING

In order to understand a process and its dimensions the process needs to be described, modelled and analysed. To be able to model the process the architecture of a process needs to be understood first and all of the elements of the process need to be taken into consideration. In this chapter the different process types, process models and main phases of process modelling and process management are described. The theoretical framework of the thesis work described in this chapter is utilised when modelling and analysing the process of course enrolment.

3.1 Process

A process may be defined as a natural phenomenon marked by gradual changes that lead to a particular result, a natural continuing activity or function or a series of actions or operations conducing to an end (Laguna & Marklund 2004, 2). A process can be seen as a set of activities which are undertaken in order to manage, develop and maintain software systems. This includes all of the techniques needed to perform the tasks, the actors who perform the activities, their roles in the process, constraints and the artifacts produced. (Acuña & Sanchez-Segura 2006, 5.)

There are many definitions for a process but the traditional high-level definition for a process more generally used is that a process simply specifies the transformation of inputs to outputs. The transformations within a process are typically classified as; physical, locational, transactional or informational. Physical transformation can be for instance the transformation of raw materials to a finished product. Locational transformation can be for instance the transportation service offered by a trucking company. Transactional transformation may include for example a banking service offered to a customer. Informational transaction on the other hand consists of for instance the transformation of data into information. The transformation perspective forms the basis of a process view according to which any organizational entity or business can be characterized as a process or a network of different processes. (Laguna & Marklund 2004, 2.)

Based on the scope of the processes within an organization the processes can be characterized into three different types. Individual processes, which are performed by separate individuals, vertical or functional processes, which are performed and contained in a certain functional unit or department and horizontal or cross-functional processes which cut across several functional units. A hierarchy exists between these different process types where a cross-functional process can be composed of a number of connected functional processes or sub processes, which in turn consists of a number of individual processes. In general any process can be broken down into one or more activities that consist of several different activities. Figure 1 below illustrates the different process types. (Laguna & Marklund 2004, 2.)

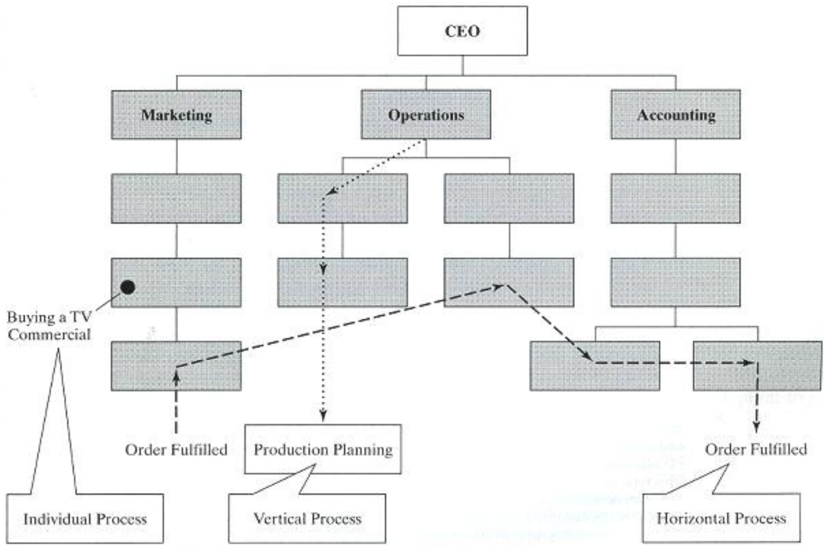


FIGURE 1. Illustration of individual, vertical and horizontal processes (Laguna & Marklund 2004, 3).

The structure of a process can be determined in the terms of its five main components: the inputs and outputs, the flow units, the network of activities and buffers, the resources and the main information structure. The inputs and outputs of a process can either be tangible, for instance raw material, or intangible, for instance information or time. The inputs and outputs define the boundaries of the process and establish the interaction between the process and its environment. A flow unit may be defined as an entity proceeding through the various activities within the process and finally exiting the process as the finished output. Flow units can include for instance materials, orders and documents. A clear understanding and definition of the process flow units is important in order to be able to design and model a process. (Laguna & Marklund 2004, 5.)

A process also consists of various activities and buffers through which the flow units have to go through in order to be transformed from input to output. This network of activities consists of all the different jobs within the process which can also be seen as processes within the process, consisting of various tasks and the relationship between these activities. Activities can be divided into two types: value-adding activities, which add value to the process or the finished output and non-value adding activities, which do not add value to the process itself or to the output of it. Buffers within a process can be seen to be storage places between the activities where the flow units pass through on their way from one activity to another. Buffers create waiting times and in this way can slow down the process. (Laguna & Marklund 2004, 6-7.)

Resources are tangible assets which are necessary to perform activities within a process. As opposed to inputs which flow through the process and transform into outputs and leave the process, resources are used within the process. The information structure of a process defines all of the information which is necessary in order for the process to function effectively. The information structure specifies which information is needed and which is available in order to make the necessary decisions for performing the activities within a process. (ibid., 8.)

According to Juran & Godfrey (2001, 3), there are three principal dimensions for measuring the quality of a process: effectiveness, efficiency, and adaptability. A process is seen to be effective if it meets the needs set for it and it is efficient when it is effective at the least cost. Remaining effective and efficient in the face of the changes that occur in the process makes the process adaptable. These three dimensions can be seen as the attributes of a good process.

3.2 Process management

Process management deals with managing, controlling and improving processes. According to Laguna and Marklund (2004, 24-26), successful process management includes three phases; initialization, definition and control. In the initialization phase the process ownership is assigned and the boundaries and the interfaces of the process defined. The objective of this phase is to find out the scope of the process and to determine who is responsible for it. In the definition phase the process is defined by understanding the process work-flow, the process activities and their relationships.

The objective of the definition phase is to document the activities and workflow which makes the process, and in doing so facilitate communication and understanding of the operational details with everyone involved in the process. In efficient product management it is not enough just to define and document the process but one must first understand the process. Also the documenting of the process needs to be understandable and clear so everyone involved in the process can benefit from it. (Laguna & Marklund 2004, 28.)

The third and final phase, controlling, deals with the issues of establishing a system for controlling the process and providing feedback to the people involved. This phase can be further divided into three different stages; establishing control points which are value adding activities, developing and implementing measures and performing feedback and control. (ibid., 31-33.)

3.3 Modelling

A model may be defined as an abstract representation of the architecture, design or definition of the process. The aim of a model is to describe, at different detail level, an organisation of the elements within a finished, ongoing or proposed process. This description of the process helps with the understanding of the process and can be used for the evaluation and improvement of it. The process model can be further analysed, validated and simulated to serve the needs of process control such as evaluation and improvement of the process in an organisation. Different process model approaches, which will be further discussed in this subchapter, may be divided into several sub-models which express different viewpoints and perspectives of the process. Some models focus on the definition of the agents involved in each activity of the process while others focus on the existing relationships between these activities. Furthermore the organisational culture may also be the focus of the model, where the behavioural capabilities or roles involved in the process are described. It is important to remember that a model is always an abstraction of the reality and as such it can only represent a partial and simplified description of the reality and not all the parts or aspects of the process can be taken into account in the model. (Acuña & Ferrè, no year of publication, Software process modelling.)

According to Acuña & Sanchez-Segura (2006) there are at least two reasons for using process models, firstly it is used to achieve better ways of defining and guiding development, maintenance and evolution processes. Secondly it used to achieve better ways of improving processes at the level of individual activities and the process as a whole. In their article, Software process modelling, Acuña and Ferrè (no year of publication) go further in listing some specific goals and benefits of modelling the process. The ease of understanding and communication of the process is an important benefit of modelling, this requires that the process to be modelled contains enough information for its representation. Modelling also provides process management support and control by helping the management to acquire a deeper project-specific understanding. Modelling provides automated orientations for the process performance by requiring an effective development environment, providing user orientations, instructions and reference material. Modelling also helps with the automated execution support by for instance requiring automated process parts and co-operative work support. A major benefit of modelling is the support which it gives to process improvement. With the help of modelling a reuse of well-defined and effective processes is possible, the comparison of alternative processes is easy and modelling offers significant support for the development of a process.

Different elements of a process, for example, activities, products, resources such as personnel and tools, and roles, can be modelled. The elements most commonly modelled are actor, role, activity and product. Actor is an entity in the process that executes an activity. Actors can be divided into two groups according to their relationship with the process, human actors and system actors. Human actors are people involved in the process and system actors are the computer software or hardware components of the process. An actor of the process can have several roles which are composed of consistent sets of activities. Role describes a set of actor or group responsibilities within the process, in other words the rights and skills required to perform a certain process activity. For example different users of an information system have different rights to perform functions within the system, such as administrators and normal users. Activity is the phase of a process which produces visible changes of the state of the process. Activity can have an input, output and some intermediate results. Activities can be performed by a human actor or by a tool used in the process. Product of the process is the material within a process, this can be either a physical product in a development process or data or information passing through the process. (Acuña & Ferrè, no year of publication, Software process modelling.)

3.4 Modelling approaches

There are several different types of modelling approaches, depending on the desired viewpoint, the process can be modelled for example at different levels of abstraction or with different goals. The types of information in a model can also be organised from different viewpoints. Common information perspectives of a process include functional, behavioural, organisational and informative perspectives. (ibid.)

Functional perspective represents the process elements which are being implemented and which information entity flows are important for these elements. Behavioural perspective describes when and under which conditions the process elements are utilised. Organisational perspective focuses on the actors and actions done by these actors in the process, in other words where and by whom in the organisation the process elements are implemented. Informative perspective describes the information entities output or the influence of the process in them, including their structure and relationships. Even though these approaches provide a vast amount of information of the modelled process none of them alone covers all the classes of information within a process, therefore most of the models used in process modelling combine one or more of these approaches. (Acuña & Ferrè, no year of publication, Software process modelling.)

3.5 Modelling notations

Modelling languages or so called modelling notations are graphical or textual presentations of a model. As well as many different types of model approaches exist there are also many different modelling notations. As in many cases the processes to be modelled are vast both in size and in contents making the modelling of them, by using only one notation difficult, therefore a combination of different notations is generally used in modelling. Modelling notations can be classified into process, data, structural, time-dependent, object-oriented and interaction language models. (Free translation from Pohjonen 2002, 62-63.)

In the process model the system is viewed from the process point of view where all the necessary actions of the process are described in a general level. This notation enables to view the possible branching and repetitions of the actions in the process but does not give actual details of the specific actions. Data model looks at the process from the structural point of view but the main focus is on the information within the process not the structure needed to process the information. Data model divides the information into logical concepts and describes the interactions between them. The most commonly used way to describe data models is an entity relationship diagram in which the interrelationships between the entities are described. Another example of a data model is data flow diagram which is a graphical representation of the information flow within the system. (ibid., 65.)

Structural models describe the overall structure of the process. Interaction models describe the interaction between the users and the system. Examples of interaction models are state model in which the system can be described both from the process and interaction point of views. It is based on the idea that each point or part of the process has a state which can change as a result of an action performed in or outside of the process. The functionality of the system is therefore based on the state transitions within the system. Sequence chart is an example of time dependent modelling where the focus is on the systems or parts of it state transitions in relation to time. Object models on the other hand look at the process from the structural point of view but unlike data models the structure of the whole system to be modelled can be taken into account. In the object model the system is described as a group of interrelated objects and each objects behaviour and information are described. (Free translation from Pohjonen 2002, 63-67.)

3.6 Unified modelling language

One example of object oriented notation is the unified modelling language (UML) which includes a set of graphical notation techniques to create visual models of software intensive systems. The diagrams used in UML represent two different views of a system, the static or structural and the dynamic or behavioural. In the static view the emphasis is on the static structure of the system, using objects, attributes, operations and relationships. This view includes the use of class diagrams and composite structure diagrams. Dynamic view emphasises the dynamic behaviour of the system by showing collaborations among objects and changes to the internal states of objects. This includes sequence diagrams, activity diagrams and state machine diagrams. (Arlow & Neustadt 2002, 3-8.)

UML will be used in this thesis work as the primary graphical notation technique to describe the existing process of course enrolment and its problems, together with the new part to be developed to solve these issues. UML diagrams used in this thesis work are the activity diagram which is used in modelling the process and the class diagram which is used to model the database used in the process. The activity diagram is a dynamic model which describes the behaviour of the system by showing the process as a collection of activities and the transitions between the activities. Activity diagram can be seen as a state chart in which each state has an entry action which specifies some process or function that occurs when this state is achieved. The most common use of activity diagram is to flowchart operations. In order for an activity diagram to give an effective model of the operations it should be focused on describing one specific aspect of a systems dynamic behaviour. (Arlow & Neustadt 2002, 232-233.)

Activity diagram contains action states and subactivity states. Action states represent the actions or in other words the tasks which cannot be divided into subtasks, which are uninterruptible and instantaneous. Action states may also be defined as states which have one entry action and at least one outgoing transition. Action states are shown in the action diagram as boxes with rounded ends. Each activity diagram has two special states, the start state and the stop state. The start date shows the beginning of the workflow while the stop state describes the end. Subactivity states on the other hand are non atomic and can therefore be broken down into other subactivity and action states, they may be suspended and may also take a limited amount of time. (Arlow & Neustadt 2002, 233-235.)

Transitions, decisions, forks and joins, object flows and signals are also methods used in the activity diagram modelling. Transitions represent the transitions out of the current state into the next state whereas decision specifies alternative paths which can be taken in the process. There can only be one input path to a decision but many output paths may exist. Forks are used to split a path into two or more concurrent flows and join in turn is used to synchronize these flows. Object flows describe how objects are input to, output from and changed by the different states. Signals represent a package of information which is communicated asynchronously between two objects. The activity diagram may also be partitioned into different swimlanes which are commonly used to represent use cases, classes, components, roles or as in the case of this thesis work organizational units. An example of an activity diagram partitioned into swimlanes is shown in figure 2. (ibid., 235-242.)

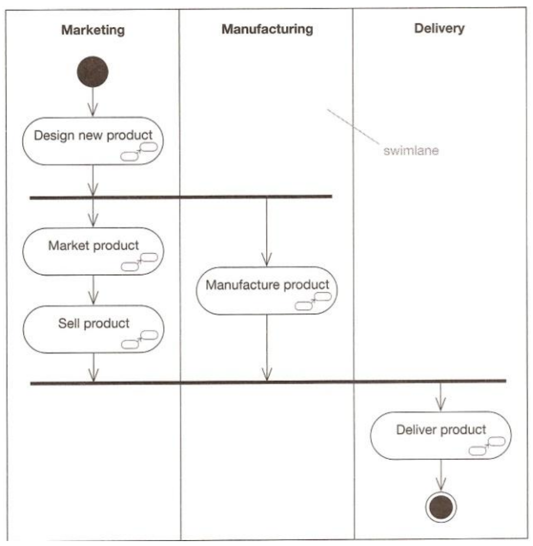


FIGURE 2. Swimlanes representing the organizational units within a business. (Arlow & Neustadt 2002, 239.)

In object oriented modelling classes, objects and their relationships are the main modelling elements. Classes and objects describe what is in the system and the relationships between them show how they are structured in relation to each other. Class diagram describes the static structure of classes in the system and their relationships. The classes represent “the things” that are handled in the system. Different kinds of relationships between the classes may exist. Associated relationship, in which the classes are connected to each other. Dependent relationship in which one class depends or uses another class. Specialized relationship where one class is a specialization of another class. Packaged relationship where the classes are grouped together as a unit. Class diagram is considered as static as the structure described is always valid in any point in the system. (Eriksson & Penker 1998, 18, 65-67.)

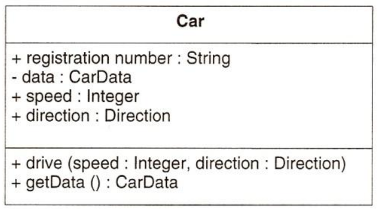


FIGURE 3. “The class Car has attributes and operations” (Eriksson & Penker 1998, 73).

The classes shown in the class diagram not only show the structures of the information in the system but also their behaviour. An example of a class is shown in figure 3 above which describes the class car where the operation drive has parameters speed and direction and the operation getData has a return type, CarData. Class is modelled with a rectangle which is divided into three compartments, the name compartment which contains the name of the class, the attribute compartment which describes the characteristics of the objects, and the operation compartment in which all the operations which are used to manipulate the attributes or perform other actions are shown. (ibid., 67-73.)

4 CURRENT PROCESS OF COURSE ENROLMENT

According to Laguna & Marklund (2004, 81) to acquire the necessary understanding of an existing process in order to improve it, it is important to know what does the existing process do, how well or poorly does it perform and what are the critical issues of the process performance. They also state that understanding the process is more important than analyzing it by documenting every detail of the process. In this part the current process of course enrolment is described and modelled and its problems defined with the help of UML modelling techniques. The new design is then introduced as the solution to these problems and the connection of it to the existing process illustrated. The main sources for this chapter were the Planning Officer of the School of Business and Information Management, Inkeri Hedemäki, Senior Planning Officer of Oulu University of Applied Sciences, Samuli Malinen and Planning Officer of Oulu University of Applied Sciences, Onno De Bruijn, who were interviewed in order to get a clear understanding of the current process of course enrolment. A study of the existing source code of Asio EduERP-student management software was also done in order to gather information of the database currently in use. The names of the actual database tables used in the current process and their attributes have been altered for security reasons.

4.1 Current process

The current process of the course enrolment for the common free choice and the free choice studies of Oulu University of Applied Sciences begins with the need to set a common free choice study course or a free choice study course available for the students. The study office creates an operation plan for a degree group by selecting the planning year, starting year of the group to which the courses are offered to, selects the degree group option to be VAP (code for the common free choice study courses) or VAT (code for the free choice studies of a specific degree program) and selects the type of a course to be offered. After this the needed courses are added to the operation plan course listing by selecting them from the plan preparation database. This is done by looking up the course codes from the online study guide according to a degree program in question and then searching them individually from the database according to the courses course code. At this point the semester when the course will be held is also checked from the online study guide and attached to the course.

When all the needed courses are added to the list they are checked through, to make sure that they hold all the correct information of the course. At this point the teachers can be attached to the courses and the maximum amount of students for the course can also be selected but this is optional. After the selections are done the information is saved into the database. The information which is saved into the database and the database are described later on in this chapter. When the operation plan is saved into the database the availability dates for the courses to be open for enrolment are decided by the study office and the request for the dates is sent by an email to the information management department.

The availability dates to be set for the courses are then hard coded into the source codes of the code packages by the information management department. This is troublesome as there are many different source code packages from which the correct date notation needs to be checked before adding the availability dates into the correct database table. After the changes are made to the availability dates the course information is shown to the students during the time of the availability dates set in the previous stage.

The courses in the student user interface are listed according to the students study unit which is checked by the system when the student signs in. The student has two separate user interfaces for viewing the courses which are available for enrolment. One for the common free choice study courses and another one for the free choices study courses enrolment. In the common free choice study user interface all of the possible courses available for the student are listed. In the free choice study user interface all of the courses offered according to the students degree programme are listed. After the student has done the course selections the information is saved into the database table *completion*, described later in the chapter.

All of the above mentioned steps can be seen in figure 4 (also in Appendix 3), where the different activities in the process are shown in an activity diagram of the current process. In the activity diagram the swimlanes consist of the information system which holds all the databases and the web interface of Oulu University of Applied Sciences, information management department, the study office and the student.

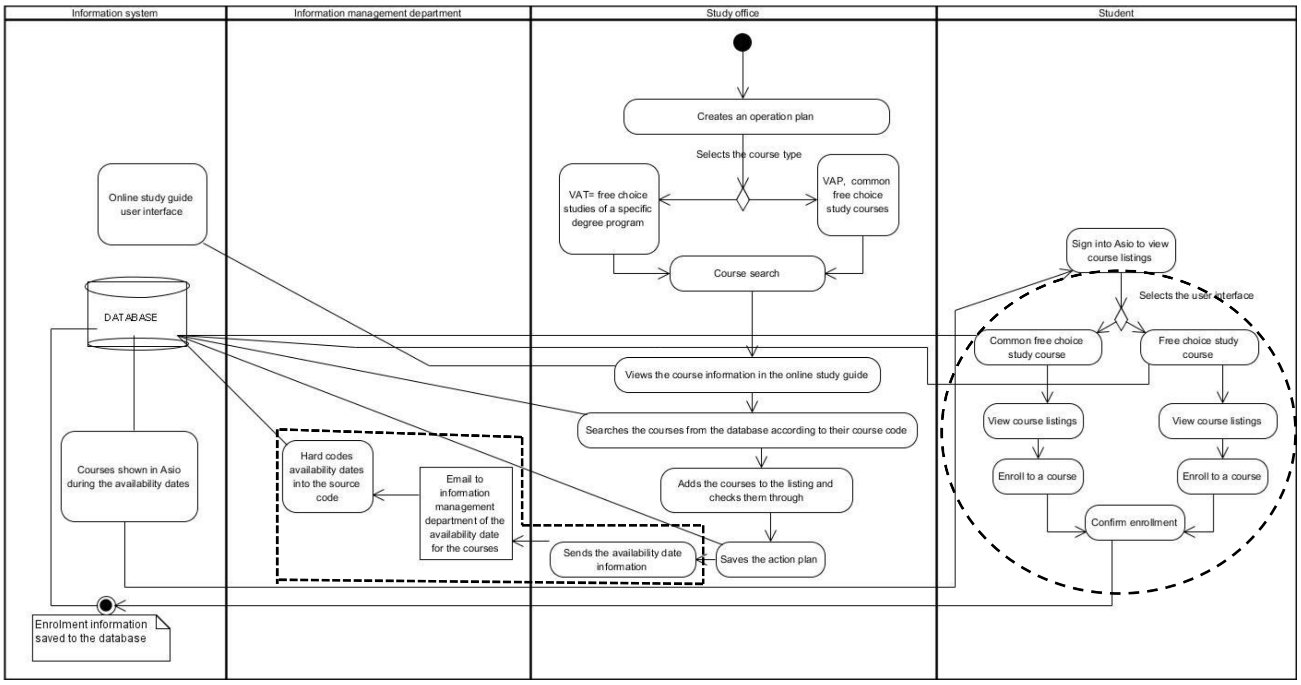

Information system
Information management department
Study office
Student
Creates an operation plan
Selects the course type
VAT= free choice studies of a specific degree program
VAP, common free choice study courses
Online study guide user interface
Course search
DATABASE
Sign into Asio to view course listings
Selects the user interface
Common free choice study course
Free choice study course
Views the course information in the online study guide
Searches the courses from the database according to their course code
View course listings
View course listings
Courses shown in Asio during the availability dates
Hard codes availability dates into the source code
Email to information management department of the availability date for the courses
Adds the courses to the listing and checks them through
Enroll to a course
Enroll to a course
Confirm enrollment
Sends the availability date information
Saves the action plan
Enrolment information saved to the database

FIGURE 4. Activity diagram of the current process.

The parts marked with a dashed line in figure 4 are those parts of the process which will be addressed in this thesis work in order to improve the current process. These consist of the two separate student user interfaces, the sending of the availability dates to be set for the courses and the hard coding of the availability dates done by the information management department.

4.2 Database of the current process

When the study office makes an operation plan for a specific class the course information such as the course number, course code, course name, credits, teacher of the course and target group of the course are saved in the Oracle database in use in Oulu University of Applied Sciences. The database tables most relevant to the course enrolment process are described in a class diagram of the database of the current process in figure 5. More database tables exist which are involved in the course enrolment process but are not seen as relevant to the process as the ones described here.

The main database tables where the operation plan information is saved are *courses_class* and *course*. The operation plan information can be found from the database with the course number, and course code. Table *courses_class* contains all the information of courses offered to a specific class according to their class specification, class code and starting year. In this table the type of the course which was specified in the making of the operation plan by the study office are classified as follows; P= basic studies, W= optional professional studies, S= compulsory professional studies, T= optional professional studies and V= optional studies. The table is connected to the *degree_program* table which consists of information of all the different degree programs offered by the university and each class belongs to a specific degree program. *Degree_program* table also holds all of the information of the different degree programs of the university and the units of the different degree programs. Abbreviations K, U and N are used for the School of Business and Information Management so these are the only values used from this column in the table.

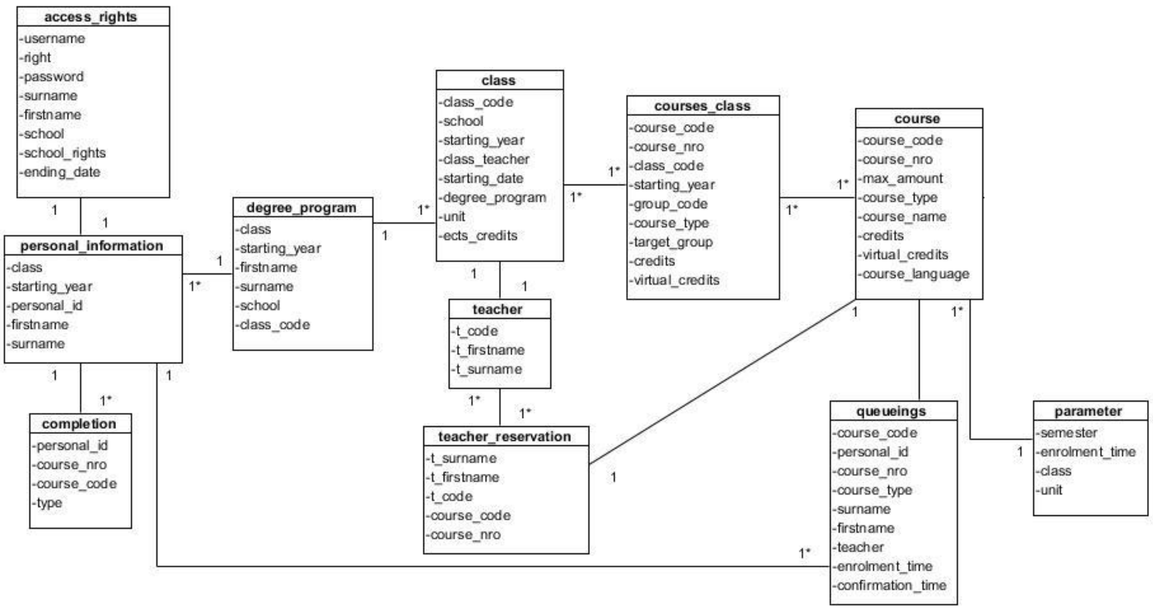


FIGURE 5. Database of the current process.

Information of all the courses offered by Oulu University of Applied Sciences can be found from table *course*. Table *class* holds information of all the classes in the university. A specific class can be searched from the database with the class code or the starting year in which case all of the classes which started on that specific year will be listed.

Teacher table includes basic information of the teachers of the university and *teacher_reservation* table consists of teacher reservation information such as to which courses a teacher is attached to. In *personal_information* table all of the student information is saved including the class of the student, starting year, degree program, unit and personal id number of the student. *Completion* table gathers all the information of students courses to which they have enrolled for or have already completed. *Queuings* is the table into which the information of a student waiting list to a course is gathered in case a course is full and more students have enrolled for it.

Access_rights table holds the access right information of the users of the system. Table *parameters* holds information of all the degree program options of the university, all the classes, semesters and the availability dates to be set for the courses. This table is the main table to which the availability dates of the course enrolments are saved to and to which the correct semester for the course is set into.

The tables most relevant to this thesis work are *courses_class, course, queueings* and *parameters*. As mentioned before the main tables into which the course information is saved when creating an operation plan for course offering are tables *courses_class* and *course*. Therefore these database tables need to be taken into account when developing the new database table for the students user interface and management of course information. *Queueings* table also needs to be taken into account when arranging the automated queuing option for the courses. However the most important database table of the current process is table *parameters* which, as mentioned previously currently holds the information of the availability dates and semesters set for the courses to be offered. As both of these parts are seen as problematic in the current process this table plays a major role in the changes to be made to the process. Nevertheless all of these tables need to be taken into account when developing the new database table for the management of course information and the user interface in order to define those database tables from where the course information is brought from and where the course information is saved to.

4.3 Problems of the current process

The main problems of the current process are all related to the student logistics part of the Asio EduERP student management software. In the current process the input of the courses for the study office works generally well even though some clear delays in the process can be seen in the part of the process where the course listing for the operation plan is done. The course codes need to be searched form the online study guide and then individually from the database according to the course code and this takes a lot of time. This problem will not be addressed in this thesis work but the main focus is on solving the issues related to the two separate user interfaces for the students and the database tables where the information is currently saved. For future development of the process the issue of the operation plan course listing is one which should be addressed, together with the option for the study office to create listings of students who have not enrolled to any courses. Currently this is done by requesting a listing from the information management department which is very time consuming for both departments.

The setting of the availability dates for the courses requires a lot of hard coding and is very time consuming. The date notations of the availability dates are different in the different code packages used in the system. This requires the administrators to remember which type of date notation is used in the different code packages. Changes to the selected semesters for the courses are also difficult to do. The changes need to be done directly into the source code to change the semester of a course and to remove the already selected one. The offering of courses which can be set to be available for enrolment is very strict in this way making the current system very static in its use for the course enrolment. Changes were seen appropriate in order to make the system accommodate a broader offering of courses. This is also important when considering the future enrolment for all courses which will be compulsory for the students. Overall the managing of course information is difficult.

The need to combine the two different student user interfaces was also seen to be important together with the option for the students to be able to view their selected eHOPS courses which will be offered for enrolment. This will help in simplifying the enrolment process for the students and in making the course selections easier. As the outlook of the current student user interface will be changed during the thesis process it was seen as a good time to also make the needed changes to the student user interface for course enrolment.

All of the problems of the current system which were found during the modelling and analysis of the current process are listed below. The problems which will be addressed in this thesis work are marked with an asterisk (*) at the end of the problem description.

Problems

- The course search done by the study office: The course search needs to be done by selecting the needed courses one by one from the database by searching them with the course code given in the study plan. This takes a lot of time as there are many common free choice and free choice courses.

- Listing of the students who have not enrolled for any common free choice or free choice courses needs to be done by the information management department as there currently is no option for listing these students, only the listing of enrolled students is possible.
- Managing of the course information is difficult: In case there are any problems with the course listings the whole source code of the packages needs to be gone through in order to find the underlying problem. This also needs to be done if changes need to be made to the course information for example to the course types which will be offered for enrolment. This makes the managing of the system very difficult. *
- Strict selection options for courses: The offering of courses which can be set to be available for enrolment is very strictly lined as the courses can only be offered according to a degree program and not for example according to an individual study group. The course offering is also limited according to a specific study unit. This makes the current system very static.*
- Hard coding of the source code: A lot of hard coding into the actual source code of the code packages is needed, in order to set the correct availability dates for the courses. The hard coding needs to be done in many different course packages and modifications to the date information in the parameters database table also needs to be done when changing the availability dates. *
- Different date notations: The used date notations are different in different source code packages and these dates need to be changed according to these formats. A lot of guess work is in play here as currently no list of which date notation is used and where exists. *
- Changes need to be done to the selected semesters: The selection of the offered semesters is also very strict and many changes need to be done directly into the source code to select one semester and to remove the other. *
- Enrolment for all courses: This would be very difficult with the existing system as it does not accommodate the option to set all of the courses available for enrolment. *
- Two separate user interfaces for the students unnecessarily complicates the student course enrolment process. *

5 MANAGEMENT AND USER INTERFACE FOR COURSE ENROLMENT

A new database table *tame* was developed together with a new student user interface (UI) in order to improve the course enrolment process and to solve some of the listed problems of the current enrolment process. The functionality of the table *tame* and the student UI, their importance to the course enrolment process and the changes they made to the current process are described in this chapter together with the underlying programming solutions of the student user interface. The names of the actual database tables used and their attributes have been altered for security reasons.

5.1 Management

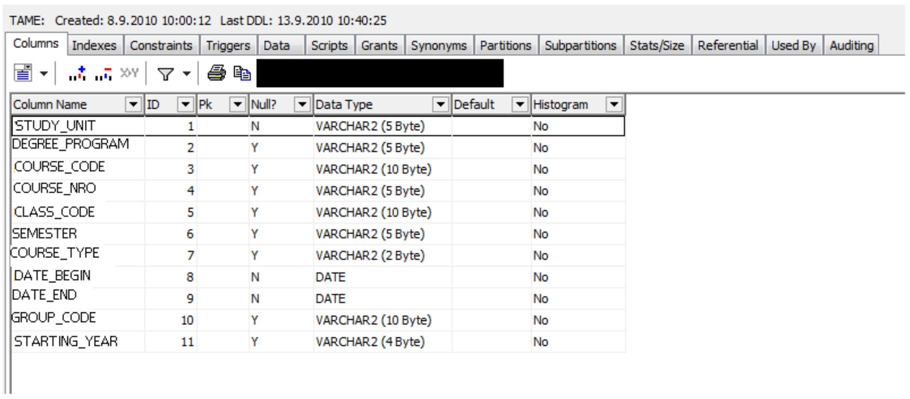
The current system of course enrolment uses many existing database tables in order to make the setting of course offering possible. The new database table *tame,* designed to improve the management of course information, will replace the existing *parameters* table which was used to retrieve the enrolment dates and semesters set for the courses. Also all of the course information which is brought to the new student UI from other database tables is filtered through the table *tame.* The available courses are only shown to the students during the availability dates set in the table *tame.*

The created select statements for the source code of the student UI will only bring course information from other database tables if the same course can be found from table *tame.* For the course selections five different select statements were used in order to bring the course information to the student UI. The used select statements were different depending on the course type in question or whether the course was found in the eHOPS of the student. Basic course information, like for example the course name, teacher, credits of the course, the target group of the course and additional course information was brought to the student UI from the database table *courses_class* by filtering the information through table *tame*.

An example of one of the select statements used can be seen below where the free choice study courses of a specific group, course type W or T are selected from tables *courses_class* and *tame*. The select statement attributes have been changed from the original ones but the functionality remains the same.

```
select a.* from courses_class a, tame b
where a.course_nro=b.course_nro
and a.course_code=b.course_code
and (b.course_type='W' or b.course_type ='T')
and sysdate between b.date_begin and b.date_end
and b.class_code=sql_class
and b.starting_year=sql_startingyear
and a.target_group=sql_class
order by a.course_code, a.course_nro
```

The database table *tame* shown in the Toad for Oracle schema browser seen in figure 6 describes all of the different attributes and the data types used in the table. Study_unit is the one letter code for the study unit, this value cannot be null. The main study unit code used in the program was K, the code for the School of Business and Information Management. Degree_ program, the three first letters of a degree program are stored into this column. TIK, BIT, DIB and KIR were the degree programs used as examples as these are some of the degree programs of the School of Business and Information Management.


TAME: Created: 8.9.2010 10:00:12 Last DDL: 13.9.2010 10:40:25

Columns | Indexes | Constraints | Triggers | Data | Scripts | Grants | Synonyms | Partitions | Subpartitions | Stats/Size | Referential | Used By | Auditing

[redacted]

Column Name	ID	Pk	Null?	Data Type	Default	Histogram
STUDY_UNIT	1		N	VARCHAR2 (5 Byte)		No
DEGREE_PROGRAM	2		Y	VARCHAR2 (5 Byte)		No
COURSE_CODE	3		Y	VARCHAR2 (10 Byte)		No
COURSE_NRO	4		Y	VARCHAR2 (5 Byte)		No
CLASS_CODE	5		Y	VARCHAR2 (10 Byte)		No
SEMESTER	6		Y	VARCHAR2 (5 Byte)		No
COURSE_TYPE	7		Y	VARCHAR2 (2 Byte)		No
DATE_BEGIN	8		N	DATE		No
DATE_END	9		N	DATE		No
GROUP_CODE	10		Y	VARCHAR2 (10 Byte)		No
STARTING_YEAR	11		Y	VARCHAR2 (4 Byte)		No

FIGURE 6. Schema browser view of the tame database table.

Course_code is the letter and number combination code given to each course. Course_nro is a number given to each course. A specific course can be found from the database with the combination of the course code and course number. Class code is the six letter code of the class, BIT7SN for example. Semester is used to set a specific course to a certain semester. Course_type of the course holds the one letter code of the course type. This can be; P= basic/ compulsory studies, S= compulsory professional studies, V= optional studies, W= optional professional studies or T= optional professional studies. Group_code can be used to combine the class code and the starting year of the class to specify a unique group. Starting_year is the two character starting year of the class. A specific class can be found from the database by using the class code and the starting year of the class.

Date_begin is the enrolment start date in the form of ddmmyyyy. Date_end is the enrolment end date also in the form of ddmmyyyy, these values cannot be null. The courses are open for enrolment from the date which is set as the date in the date_begin until the date set in the date_end column. For example if a course needs to be open for enrolment from the 1.9.2010 until the 19.9.2010 the dates set in the database table need to be; date_begin = 1.9.2010 and date_end = 20.9.2010.

By using table *tame* each course to be offered for enrolment can be set a specific enrolment time and no hard coding into the source code of the new student UI is needed as the availability dates set for each course are brought directly from table *tame.* The date notation used in the table is the same as in the source code of the student UI, this way solving the problem of the different date notations used in the current system.

The option to offer a course for all of the units of the university is made possible by setting the study unit code for the course to be an asterisk (*) which is noted in the source code of the UI to be open for all the different study units of the university. This makes it possible to offer any course for enrolment for all the units of the university. This helps with the current systems problem of strict selection option for courses to be offered by making the course offering more flexible and manageable.

The courses can also be offered for enrolment according to a specific degree program, individual study group or even by according to the status of the course as it is done in the new student UI. The flexible course offering also helps in making the system less static and the course offering is not as strictly lined as in the current system. *Tame* accommodates the option to enrol for all courses which are available for enrolment, including the compulsory courses of a study group. As the course status in the table can be set to be any one of the course types including P= basic studies or S= compulsory professional studies, the listing of all types of courses is possible. The compulsory courses are also listed as one selection option in the new UI described later in this chapter.

Managing of the system is made easier by filtering the offered courses through *tame* table. With the help of the user interface for the database table *tame*, which should be developed in the future, the management of the system and the courses to be offered will be very easy. In case of any problems with the course listings the problems can easily be traced as the used select statements play such an important role in the changed process. Class diagram of the changed process can be seen in figure 7 where the most relevant database tables and their attributes are described.

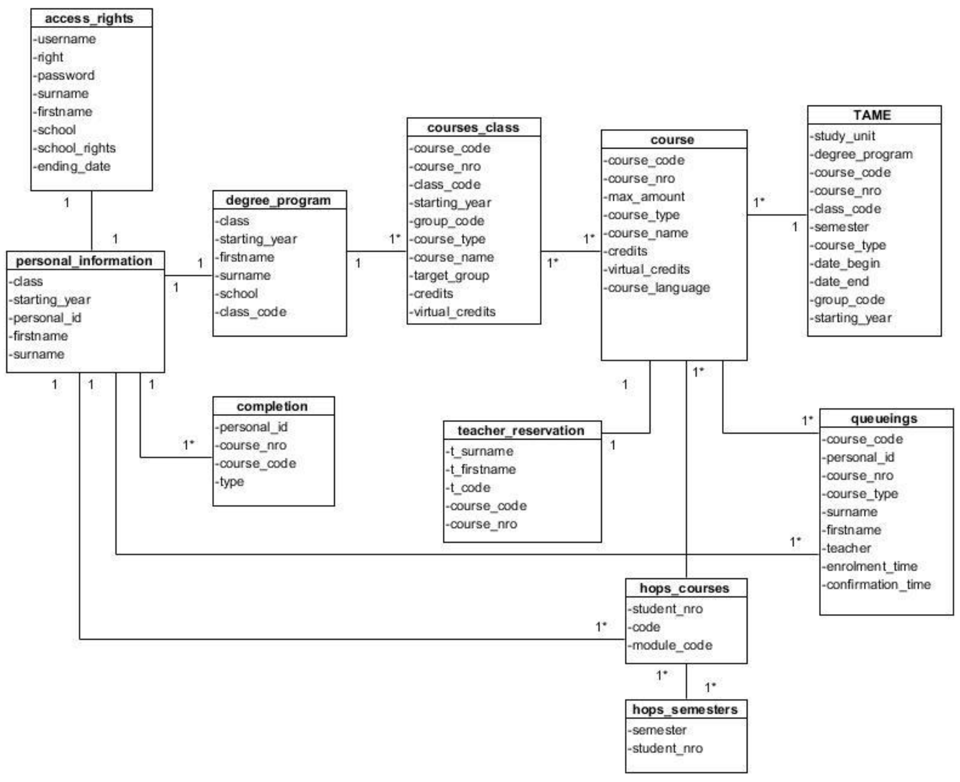


FIGURE 7. Database of the changed process.

Table *access_rights* is used to check the usage rights of the student as they sign into Asio. *Personal_information* table is used to get the students name, class, starting year of the class and personal id. *Completion* table holds the information of those courses that the student has already enrolled for or has completed. This table is also used to print out the names of the students already enrolled for a specific course. Table *degree_program* is used to find out the school of the student and in this case the school value needs to be K, U or N which are the abbreviations for the School of Business and Information Management as the UI will only be used in this unit. *Courses_class* table is used to fetch most of the course information displayed for the student in the UI such as the course id, course name, period when the course is offered, course type, course credits, virtual credits and target group.

Course table is used to get the maximum amount of students per course to be offered in the course listing of the UI and also in case if the student selects to view more information of the course. In the listing for more course information most of the attributes of the course table are needed. From *teacher_reservation* table the teacher for the course to be offered can be found and the teachers surname displayed in the course listing UI. *Queueings* table is used in the UI for viewing student enrolments and checked whether the student is queuing for any courses. This table is also used in case the course to which the student wishes to enrol for is full after which the student is automatically put into the queue for the course. Table *hops_courses* holds all of those courses which the student has selected into their eHOPS and is checked when displaying the courses in the UI to see if the courses to be offered exist in the table *hops_courses*.

Table *hops_semesters* is used to check the semester for which the student has selected the course for in their eHOPS. As mentioned before *tame* table is used to filter all of the courses which need to be put available for enrolment. The courses beginning and end enrolment dates are brought to the user interface from this table and the course is available for enrolment only during those days.

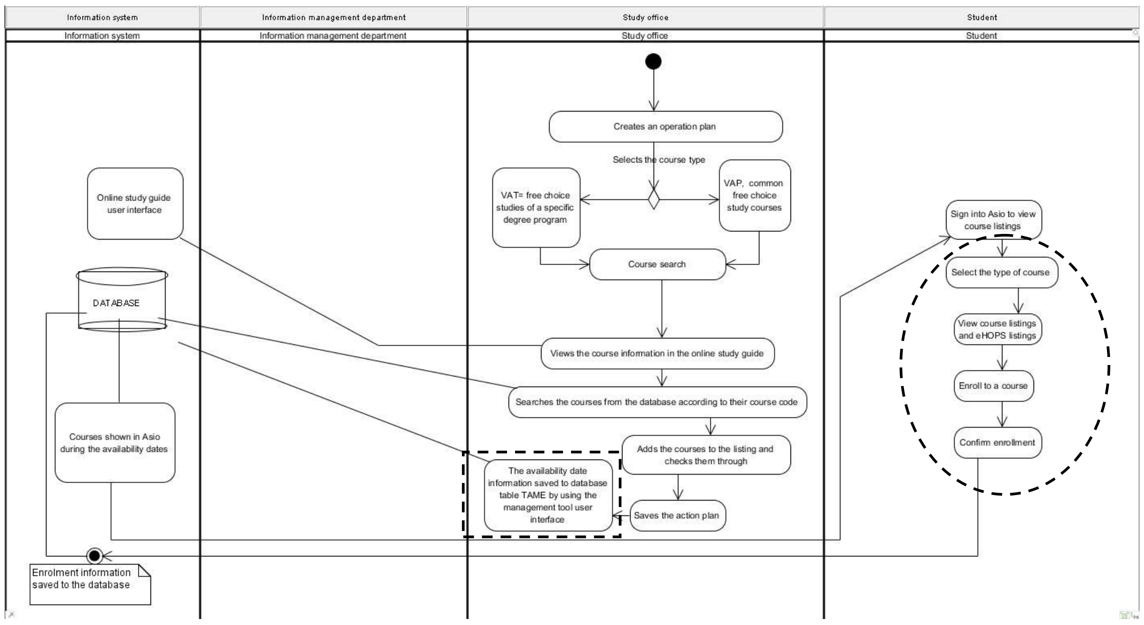

Information system
Information system
Information management department
Information management department
Study office
Study office
Student
Student
Creates an operation plan
Selects the course type
VAT= free choice studies of a specific degree program
VAP, common free choice study courses
Course search
Online study guide user interface
DATABASE
Views the course information in the online study guide
Searches the courses from the database according to their course code
Adds the courses to the listing and checks them through
Saves the action plan
The availability date information saved to database table TAME by using the management tool user interface
Courses shown in Asio during the availability dates
Enrolment information saved to the database
Sign into Asio to view course listings
Select the type of course
View course listings and eHOPS listings
Enroll to a course
Confirm enrollment

FIGURE 8. Activity diagram of the changed system.

The activity diagram of the changed system described in figure 8 shows the changes in the current system, marked with a dashed line, which occur after the database table *tame* and the new student UI are taken into use. All of the improvements in the process made by database table *tame* are fully realised only after the actual implementation of the user interface for the table. The UI will enable the study office to set the courses to be available according to their own specifications. This will leave out the phases of sending the availability dates to be set for the courses and the hard coding of the availability dates done by the information management department.

5.2 Student user interface

The new student UI combines the previous two separate user interfaces, the common free choice study courses and the free choices study courses currently used for course enrolment. It also gives the student the option to view the courses which they have enrolled to and the courses which they have selected into their eHOPS and are offered for enrolment.

As it can be seen from the activity diagram of the changed system described in figure 8 the student user interface has been simplified by removing the two separate user interfaces for enrolment by creating one user interface for the students. In this UI the students are able to select the course types to which they wish to enrol for. The compulsory courses are one selection option and also the students own enrolments and the option for their removal is given.

The courses are only shown to the students if the enrolment date of the course is correct as the UI checks if the current date is between the enrolment dates of the course set in the database table *tame*. By using *tame* table any type of course can be set for enrolment for the students like the compulsory courses of the degree program or the courses of an individual group. Before this was not possible as the course types which could be set for enrolment was very strictly lined. The course offering can also be done according to any study unit of the university.

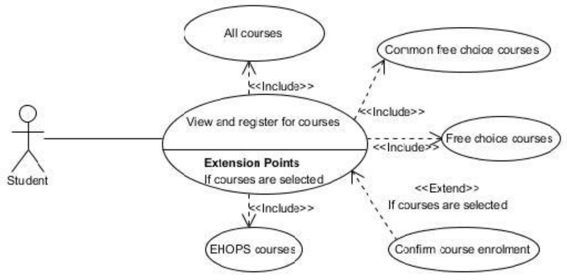


FIGURE 9. Student use case diagram.

Student use case diagram of the new student user interface is described in figure 9. The actions which have changed from the previously used UI are the option to enrol to all offered courses by using one user interface and the option to view personal eHOPS course selections. The free choice study courses of the unit of the student and the free choice study courses of the students group listings support the option to view the courses which the student has selected into their personal eHOPS.

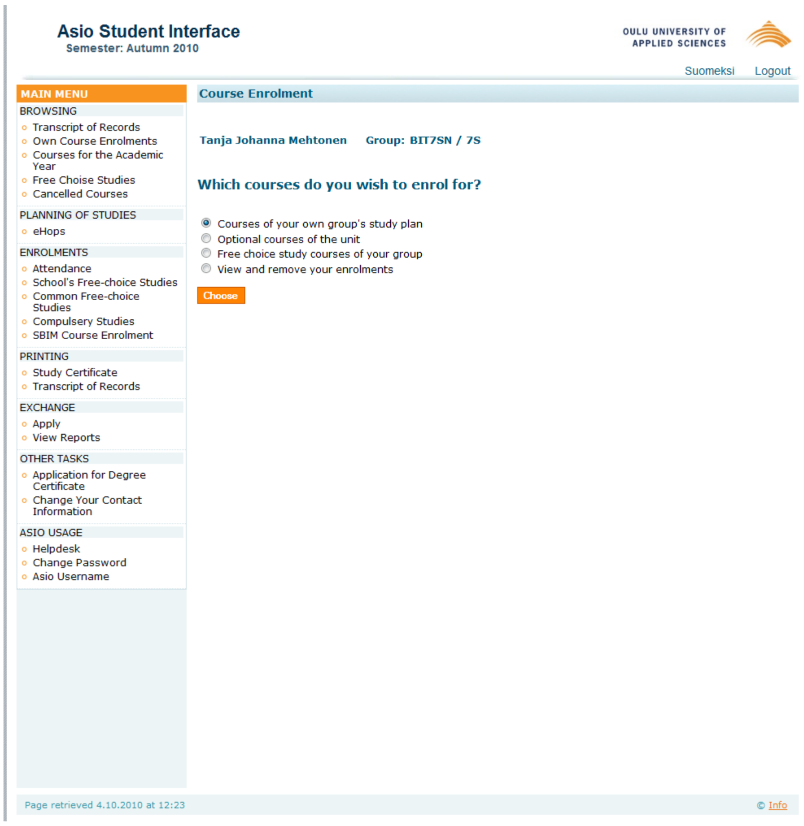
5.3 Underlying programming solutions

For the new student user interface a PL/SQL package called *students*, its specification and body part were created. In the package the following procedures and functions can be found; procedure *ehops_listing*, procedure *student_enrolments*, procedure *student_own*, procedure *main*, function *lang* and function *authorize*. The names of the packages have been changed for security reasons. Procedure *ehops_listing* lists all of the courses which the student has selected into their eHOPS and which are available for enrolment. Procedure *student_enrolments* is used when the student wishes to view their own enrolments or to remove them. Procedure *student_own* is used to display the main tables and course information used in the course listing for the compulsory courses of the group of the student, free choice study courses of the unit of the student and the free choice study courses of the group of the student. Also the abbreviation explanations of the tables, which are listed at the bottom of each page, are brought from this procedure.

Procedure *main* forms the main page of the user interface shown in figure 10. From this procedure the other procedures are called depending on the course selection done by the student. Function *lang* is used for the two different language options of the UI, Finnish and English. Function *authorize* is used as the student is directed to the main page of the UI from the main page of Asio Student Interface. Even though the student is not required to re-enter the username and password which they enter when signing into Asio they are checked when entering the main page of the UI.

The new student user interface uses the new styles and layout settings set for Asio student software. The main page of the new student UI for course enrolment can be accessed from the main page of Asio Student Interface by clicking a link for SBIM (abbreviation for the School of Business and Information Management) Course Enrolment or Liiketalouden yksikön opintojaksoilmoittautuminen, in Finnish.

On the main page of the UI the name and study group of the student are printed out and there are four radio button options for the student to select from. This page is the procedure *main* in the *students* package. The selection options include the courses of the student's own group's study plan (course_type P or S in table tame), free choice study courses of the student's study unit (course_type V), free choice study courses of the student's group (course_type W or T) or view and remove the student's own enrolments. The main page is only displayed to the students of the School of Business and Information Management as the unit is used as a case study in this thesis work. For students belonging to another study unit a text "The enrolment is not available in your study unit." will be printed out instead of the main page.



Course Enrolment

Tanja Johanna Mehtonen   Group: BIT7SN / 7S

Which courses do you wish to enrol for?

[✓ Courses of your own group's study plan]
Optional courses of the unit
Free choice study courses of your group
View and remove your enrolments

Page retrieved 4.10.2010 at 12:23

FIGURE 10. Main page of the new student user interface.

Depending on the selection that the student makes the next page is displayed. The values of the radio buttons are used in order to specify which select statement is used to retrieve course information from the database tables. All of the other selections are directed to the procedure *student_own* apart from the view and remove the student's own enrolments option, which in turn is directed to the procedure *student_enrolments*.

The course information in the course listing for the student is brought from three different database tables and by using table *tame* as the filter for returning the correct courses. Table *courses_class* is used to bring course information of those courses which are set to be available according to their enrolment dates in table *tame*. This course information includes details of the course such as the name of the course, credits of the course and target group of the course. Table *teacher_reservation* is used to bring the teacher information of the courses and the surname of the teacher is printed to the course listing table.

Only those courses which can be found both from table *tame* as well as from table *courses_class* are brought to the listing. The current date is checked to be between the enrolment dates of the courses in *tame* and only those courses which are available for enrolment are brought to the listing. Also the course type is checked to match the selection of the student and depending on the selection the correct select statement is used to bring the course information needed. Examples of these select statements can be seen later on in this chapter. *Tame* table is also used to bring the availability dates of the specific course to the course listing.

The new UI for course listing of free choice study courses of the student's study unit can be seen in figure 11. There are two separate html tables in the user interface into which the course information is brought to with a checkbox option for selecting the courses. In the first table all courses which are available for enrolment and the student has selected into their eHOPS are listed. If the courses can be found both in tables *tame* and *hops_courses* the procedure *ehops_listing* is called. The second table lists all the other available courses which the student has not selected into their eHOPS, or the student has not enrolled to or completed already. This is done in the procedure *student_own*. These two tables are used both in the course listing for the optional courses of the unit and the free choice study courses of the students own study group. In the first radio button option, the courses of the students own group's study plan, the courses brought to the listing are not checked against the courses in the eHOPS of the student as this was not seen necessary in case of compulsory courses.

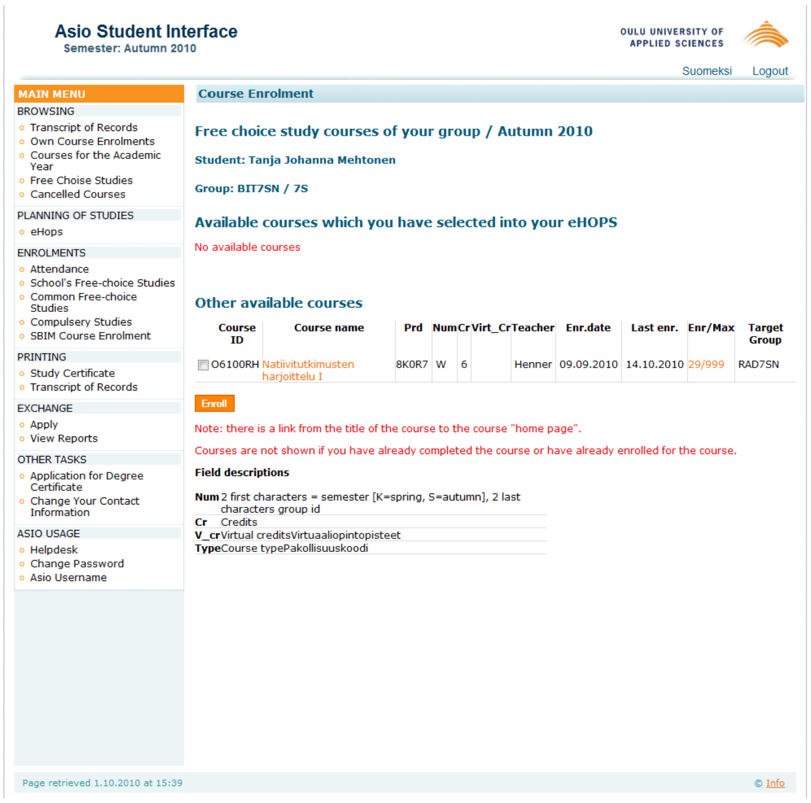
The source code for checking if the course offered for enrolment is also in the eHOPS of the student can be seen below. If the course can be found from table *hops_course* the object sql_cinehops gets a value greater than 0 and the course is shown in the procedure *ehops_listing*.

```
begin
   loop
    select count (*) into sql_cinehops
        from hops_course
        where student_nro=uid
        and code=luok.course_code;
   exit;
   end loop;
   exception when others then sql_cinehops:='0';
end;
```

The following piece of source code checks if the student has already enrolled for, or has already completed the course which is offered. The table *completion* is called and courses found with the personal id of the student are checked against the courses in the cursor which is fetched into a record structure luok. If the course is not found the sql_cfound value remains 0 and the course is listed to the student, otherwise the course will not be shown.

```
    begin
         sql_cfound:='0';
           select count(*)into sql_cfound
            from completion
            where personal_id=uid
             and
      substr(course_nro,1,2)=substr(luok.course_nro,1,2)
              and type='X'
              and course_code=luok.course_code;
           exception when others then sql_cfound:='0';
      end;
```

In case there are no courses available for enrolment (the sql_cfound !=0) or if there are no courses available which the student has selected into their eHOPS, a text “No available courses” will be displayed to the student instead of the table columns which can be seen in figure 11 displaying the course listing for optional courses of the unit. Underneath the two html tables an enrolment button is displayed. The button takes the value of the checkbox which the student has selected. The explanation of the different abbreviations used in the table columns are displayed in a table at the bottom of the page.



Course Enrolment

Free choice study courses of your group / Autumn 2010

Student: Tanja Johanna Mehtonen

Group: BIT7SN / 7S

Available courses which you have selected into your eHOPS

No available courses

Other available courses

Course ID	Course name	Prd	Num	Cr	Virt_Cr	Teacher	Enr.date	Last enr.	Enr/Max	Target Group
☐ O6100RH	Natiivitutkimusten harjoittelu I	8K0R7	W	6		Henner	09.09.2010	14.10.2010	29/999	RAD7SN

Note: there is a link from the title of the course to the course "home page".

Courses are not shown if you have already completed the course or have already enrolled for the course.

Field descriptions

Num	2 first characters = semester [K=spring, S=autumn], 2 last characters group id
Cr	Credits
V_cr	Virtual creditsVirtuaaliopintopisteet
Type	Course typePakollisuuskoodi

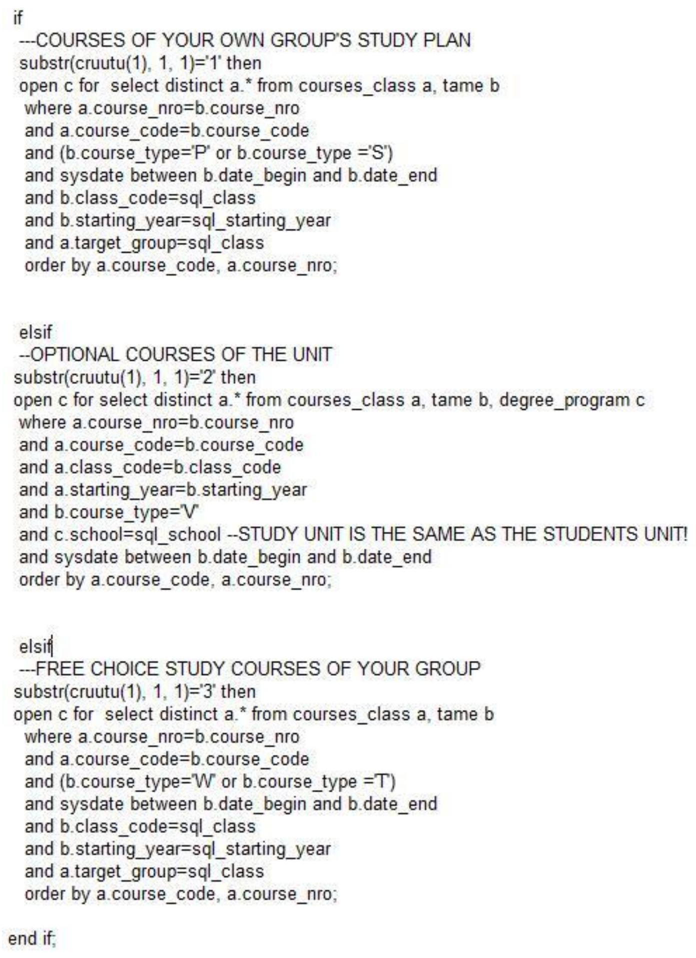
Page retrieved 1.10.2010 at 15:39

FIGURE 11. Course listing of optional courses of the unit.

After the enrolment button is pressed the student is directed away from the developed *students* package and into already existing packages of Asio where the confirmation of the enrolment is done and the enrolment information saved into the database table completion.

Cursor structures were used in the source code of the new student UI in order to handle all of the different select options for the course retrieval from the database table *tame*. In order to be able to return more than one value with an SQL statement a cursor needs to be used. Cursor is simply a construct used in order to hold the data rows which an SQL query returns. There are two types of cursors, implicit and explicit. Implicit cursors can return only one value when explicit cursor may return two or more rows but could also return 0 or only one row. (Rob, Coronel & Crockett, 2008. 458- 459.)

Because of the ability to return two or more rows explicit cursor it was used in the thesis work. These cursors were declared in the beginning of the *student_own* package which displays the listing of the available courses according to the student selection. The radio button value was used in order to decide which cursor was used and opened when arriving to this source code package. The cursor used for the three different select statements depending on the type of course selected can be seen below. Before the cursor selection the student information is brought from table *personal_information* as the unit, class, and starting year of the student are needed in the cursor select statements. The radio button option selected by the student (cruutu) is used as the condition in the if statement.



```
if
 ---COURSES OF YOUR OWN GROUP'S STUDY PLAN
 substr(cruutu(1), 1, 1)='1' then
 open c for  select distinct a.* from courses_class a, tame b
  where a.course_nro=b.course_nro
  and a.course_code=b.course_code
  and (b.course_type='P' or b.course_type ='S')
  and sysdate between b.date_begin and b.date_end
  and b.class_code=sql_class
  and b.starting_year=sql_starting_year
  and a.target_group=sql_class
  order by a.course_code, a.course_nro;


 elsif
 --OPTIONAL COURSES OF THE UNIT
substr(cruutu(1), 1, 1)='2' then
open c for select distinct a.* from courses_class a, tame b, degree_program c
 where a.course_nro=b.course_nro
 and a.course_code=b.course_code
 and a.class_code=b.class_code
 and a.starting_year=b.starting_year
 and b.course_type='V'
 and c.school=sql_school --STUDY UNIT IS THE SAME AS THE STUDENTS UNIT!
 and sysdate between b.date_begin and b.date_end
 order by a.course_code, a.course_nro;


 elsif
 ---FREE CHOICE STUDY COURSES OF YOUR GROUP
substr(cruutu(1), 1, 1)='3' then
open c for  select distinct a.* from courses_class a, tame b
  where a.course_nro=b.course_nro
  and a.course_code=b.course_code
  and (b.course_type='W' or b.course_type ='T')
  and sysdate between b.date_begin and b.date_end
  and b.class_code=sql_class
  and b.starting_year=sql_starting_year
  and a.target_group=sql_class
  order by a.course_code, a.course_nro;

end if;
```

Implicit cursors are created when using a select statement with into or bulk into clauses or the select statement is embedded inside a cursor for loop statement. The implicit cursors can be divided into three different types, the implicit bulk collection cursor, the single-row and the multiple-row implicit cursors. Both the single-row and the multiple row implicit cursors use SELECT statements. (McLaughlin, 2008. 139-142.)

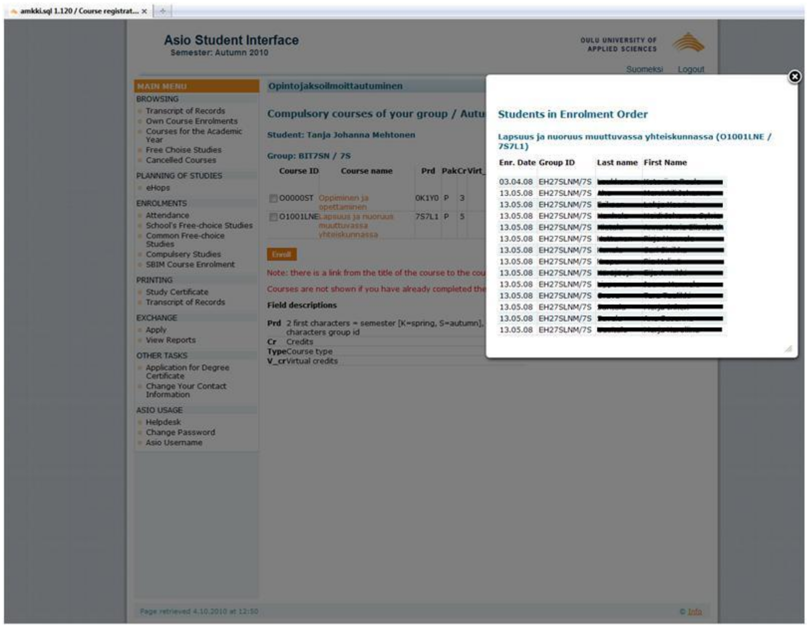
The single-row implicit cursors only work when a single row is returned by a select statement and therefore could not be used in the thesis work. Multiple-row implicit cursors can be created in two ways, either by writing any data manipulating language such as insert, update and select statements in the PL/SQL block. Or by writing an embedded query in a cursor FOR loop instead of defining the cursors in the declaration block of the source code. Multiple-row implicit cursors can also be defined inside cursor FOR loop statements. (McLaughlin, 2008. 139-142.)

An explicit cursor is created when it is defined inside a declaration block. Explicit cursors can be static and dynamic SELECT statements. Static SELECT statements return the same query each time sometimes with different results and do not change their behaviour unlike dynamic SELECT statements which run different queries each time, depending on the parameters given to the statement when it is opened. Explicit cursors always require the use of its four components, the statements for opening, fetching, and closing the cursors whether a simple or WHILE loops or cursor FOR loop statements are used. The OPEN statement is used to open cursors, the FETCH statement to fetch records from the cursors, and the CLOSE statement in order to close and release resources of the cursors. With cursor FOR loop statements you are not required to use the statements as the FOR loop opens, fetches, and closes the cursors. (ibid,139-142.)

An example of the cursor statements used in the *student_own* procedure can be seen in the following source code where the cursor is defined, fetched as a row into a record structure, a loop is iterated through with the exit condition when the cursor is no longer found, closed and a further condition is given in case the cursor is empty, in other words when there are no rows to be returned a simple text is displayed. The program exits when there are no more records to return. As the cursor is already opened in the cursor definition described in the previous piece of source code, the open statement is not done here. The statements are picked up from the source code and do not exist as such in the actual source code package.

```
TYPE cur_typ IS REF CURSOR;
   c cur_typ;
 loop
 FETCH c into luok;
 EXIT WHEN c%NOTFOUND;
 end loop;
 close c;
 if c%found then
 /* HERE THE COURSES ARE LISTED IN A TABLE*/
 end if;
 if c%notfound and lang ='f' then
 htp.p('<p class="highlight"> Ei valittavissa olevia
 opintojaksoja</p>');
 elsif c%notfound and lang ='e' then
 htp.p('<p class="highlight"> No available courses</p>');
end if;
```

In the course listing a highslide viewer was used in order to bring up the names of those students who have already enrolled for the course in a new popup window, which appears on top of the existing window. This feature was also used in the course information option where a similar window was displayed for further course information if the course name on the listing was clicked. Highslide is a JavaScript based image, media and gallery viewer, which opens the content within the active browser window (What is highslide JS. Highslide JS, 2010, date of retrieval 15.10.2010). An example can be seen in figure 13 below where a popup window of the students enrolled for a course in enrolment order is viewed. An example of the syntax used in the source code of the UI can be seen below.



Asio Student Interface
Semester: Autumn 2010

Opintojaksoilmoittautuminen

Compulsory courses of your group / Autu

Student: Tanja Johanna Mehtonen

Group: BIT7SN / 7S

Course ID	Course name	Prd	Pak	Cr	Virt_
O0000ST	Oppiminen ja opettaminen	OK1YO	P	3	
O1001LNE	Lapsuus ja nuoruus muuttuvassa yhteiskunnassa	7S7L1	P	5	

Note: there is a link from the title of the course to the cou

Courses are not shown if you have already completed the

Field descriptions

Prd	2 first characters = semester [K=spring, S=autumn], characters group id
Cr	Credits
Type	Course type
V_cr	Virtual credits

Students in Enrolment Order

Lapsuus ja nuoruus muuttuvassa yhteiskunnassa (O1001LNE / 7S7L1)

Enr. Date	Group ID	Last name	First Name
03.04.08	EH27SLNM/7S	[redacted]	[redacted]
13.05.08	EH27SLNM/7S	[redacted]	[redacted]
13.05.08	EH27SLNM/7S	[redacted]	[redacted]
13.05.08	EH27SLNM/7S	[redacted]	[redacted]
13.05.08	EH27SLNM/7S	[redacted]	[redacted]
13.05.08	EH27SLNM/7S	[redacted]	[redacted]
13.05.08	EH27SLNM/7S	[redacted]	[redacted]
13.05.08	EH27SLNM/7S	[redacted]	[redacted]
13.05.08	EH27SLNM/7S	[redacted]	[redacted]
13.05.08	EH27SLNM/7S	[redacted]	[redacted]
13.05.08	EH27SLNM/7S	[redacted]	[redacted]
13.05.08	EH27SLNM/7S	[redacted]	[redacted]
13.05.08	EH27SLNM/7S	[redacted]	[redacted]
13.05.08	EH27SLNM/7S	[redacted]	[redacted]

FIGURE 13. Iframe of the students in enrolment order.

```
htp.p('
<td class="bb bl left"><a class="noul"
href="/pls/asio/asio_kotis.kurssin_opiskelijat?tun='||luok
.course_code||'&amp;nro='||luok.course_nro||'&amp;lang='||
lang||'"
    onclick="return   hs.htmlExpand(this,   {   objectType:
''iframe'', width: ''500'', wrapperClassName: ''titlebar''
} )">'||sql_ilmlkm||'/'||sql_maxkap||'</a></td>
');
```

The number of students enrolled for a course and the maximum amount of students for a course is printed in the html table column <td>. The hs.htmlExpand function is used to open an expander with html content, in this case iframe content and a content wrapper titlebar is used. The onclick event-handler calls the address given in the href target. As the iframe is an external page the object width needs to be given.

6 DEVELOPMENT PHASES

The waterfall model, also called the classic life cycle model gives a systematic sequential approach to software development. In the waterfall model the software development process goes through the following sequential flow of phases; requirements, analysis, design, implementation, integration and deployment and maintenance. This model is used as the basic structure of the thesis work process and the different tasks of these phases are described in this chapter. (Pressman 2005, 79.)

The software to be developed is an Oracle based database which will gather needed information of the courses offered by Oulu University of Applied Sciences and provide a new user interface for the students. The software enables the staff to add, remove and modify course availability and course information, which is added into the new database table. The possibility to set compulsory courses for enrolment is also taken into account. This database table for the course enrolment process will not have its own user interface but is instead used by the administrators by utilizing the Oracle-software currently in use in the information management department for the administration, modification and development of the databases of Oulu University of Applied Sciences. The main focus is to design a new database table into which the needed information from the course enrolment process will be gathered. In this way making the setting of the availability dates and other course information for the common free choice study courses and the free choice study courses of Oulu University of Applied Sciences more dynamic and administrable.

The aim is also to create a new user interface (UI) for the students, enabling them to register to all the offered courses using the same user interface. As mentioned before currently the course enrolment is done by using two separate user interfaces, one for the common free choice study courses and another one for the free choices study courses. In addition to this the software should also support the eHOPS course viewing option, which means that the students would be able to view all of the courses they have selected into their electronic personal study plan, eHOPS, at the beginning of their studies.

Only those courses which have been selected to be offered for the students will be brought from the students eHOPS selections to the eHOPS course listing shown in the new student UI. These courses will be shown in the UI together with the offered courses. This would enable the students to take into account the choices they have previously made in their personal study plan and would hopefully make the course selection for the students clearer, as the students would be able to see which one of those courses they have selected into their eHOPS will be offered.

During the thesis work a decision was made in which it was stated that in the future the students will need to register to all of the courses offered by Oulu University of Applied Sciences. Also the compulsory courses should be listed in the new student user interface and this needs to be taken into account when specifying the requirements for the software. The School of Business and Information Management will be used as a test case when implementing this software. The planned and actual schedule of the thesis process can be found in Appendix 1.

6.1 Requirements and analysis

The requirements and analysis phase of the thesis project was done according to the schedule with the main purpose of specifying the requirements of the software to be developed. During this phase the requirements were discussed with the commissioner and requirements specification document was created. The table of contents of the requirements specification can be seen in Appendix 4.

This phase of the process is the most important one as the requirements for the software need to be fully understood in order to be able to create a system which meets the needs of the commissioner. In case the requirements are not fully understood the process will phase difficulties in the implementation phase.

The main users of the software are the students and staff of Oulu University of Applied Sciences. For the administrator the activities of the software include adding course availability dates, setting semesters for the courses and setting the groups to which the course is offered to. For the student the main activities include the ability to view available courses, viewing selected eHOPS courses and enrolment for available courses. The functional requirements were also specified with the help of use case diagrams and textual descriptions of the use cases. A student use case diagram of the new student UI can be seen in figure 9 in the user interface subchapter of chapter five. The information content of the software was described with the help of a class diagram of the basic structure of the system seen in figure 14 below.

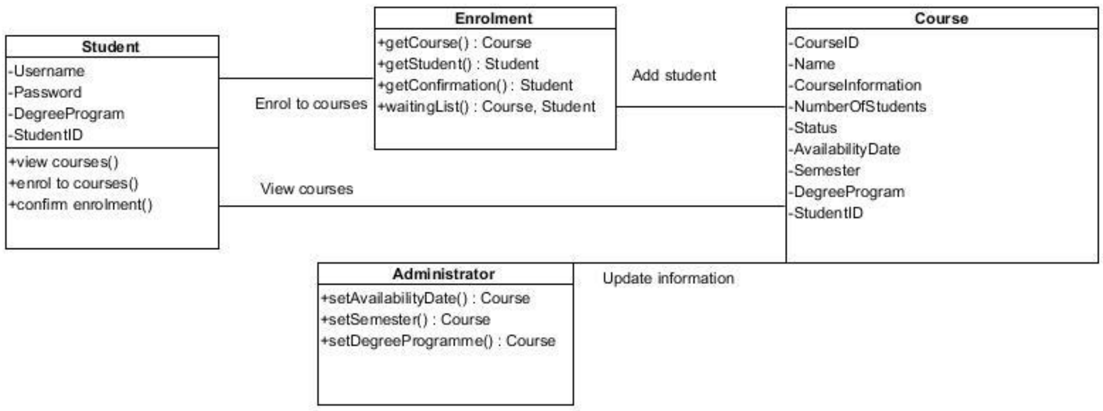


FIGURE 14. Class diagram of the system.

Non-functional requirements of the system were specified to include data reliability, maintainability, usability, security, functionality and recovery. As all the data is stored and run from the same database of the system the data integrity of the system is easy to maintain. Also in case of a system failure all the data is stored into the test side of the server where it will be implemented and tested before deployment, making it separate from the system used daily in the university. This will also help with the recovery of the data.

Maintainability requirements for the system were high in order to make defect correction, meeting new requirements and future maintenance of the system as easy as possible. Because the environment for the system will probably be changed in the future maintainability was important as the system needed to cope with the change. This is why the design of the database table needed to be as dynamic and logical as possible. Usability of the student user interface was also seen important and this needed to be taken into account in the design phase.

Security of the system would be automatically quite high as it would be a part of the existing course enrolment system of the university. Functionality of the system needed to be high and this should also be into account in the design and implementation phase of the database table and the new student user interface. The standardization of data was also discussed as Oulu University of Applied Sciences uses the standardization specifications of educational data set by the Finnish virtual university. Therefore it is important that these specifications are also used when handling data in the new software in order to make the overall system as coherent as possible.

The current course enrolment process was modelled at this phase. Both the study office and information management department were interviewed in order to create a picture of the process as a whole. After this the process was modelled with the help of an activity diagram. The diagram was reviewed by the commissioner and some minor changes were made. The parts of the process which the thesis work would change were identified. Modelling the database used in the current process was more time consuming. The source code packages used in the current process needed to be read through in order to create a list of the used database tables. After this the functionality of these tables needed to be found out and after this the database could be modelled with the help of a class diagram.

6.2 Design

Lauesen (2005, 6-7) lists the most important quality factors in IT systems in addition to usability to include correctness, availability, performance and maintainability. According to these factors the system needed to be designed in a way that there would be as few errors as possible and that the system would be available for use at any time. The performance of the system needed to be fast and the security of it high. In addition to the usability the maintainability of the system was seen as an important factor as the aim was to improve the maintainability of the current course enrolment system.

The design phase started after the requirements specification was done, which helped the design phase as the main software infrastructure was already known. First steps in the design phase were to study the existing source code packages of Asio EduERP-student management software in order to determine which source code packages were used in different stages of the current enrolment process. It was also important to clarify what kind of information was brought from the different tables to the existing user interfaces and which information was saved and where. Although it was very time consuming as the source code of each used package needed to be read through in order to determine the uses of the different database tables in them but as the most relevant database tables of the current enrolment process were already known this part of the design phase was quite simple.

Knowing the functionality of the database tables helped in deciding which of these tables should be used to bring needed information into the new student UI and where the enrolment data should be saved into. It also helped when deciding the attributes for the database table *tame* in order for it to accommodate both the filtering of the course information and the student UI and for it to be as dynamic as possible.

The main functions of the new student UI were designed together with the new layout. The new layout options for Asio, which were taken into use during the thesis process, were used in the design. The aim of the user interface design was to make the new user interface to have as high usability as possible. One way to ensure this was to design the user interface to be as simplified as possible and to only include the most necessary functions in it as needed. It was decided that the main page of the user interface would only include four different options for the students to select from. The compulsory courses of the students group, free choice study courses of the students unit, free choice study courses of the students group and view and remove students enrolments. After this the implementation of the detailed design in PL/SQL code began.

6.3 Implementation

The implementation phase progressed fast as the author could purely concentrate on the coding and testing of the software for approximately 40 hours a week. The software used during the implementation phase was Toad for Oracle 9.1. tool. When the implementation phase began the author had previous experience of PL/SQL and Toad for Oracle tool from the practical training done for the commissioner. However as it had been such a long time since using the program and PL/SQL it took a while to get used to them again. The commissioner was very supportive during the whole implementation phase.

The first task of the implementation phase was to develop the database table *tame* for the management of course information. The table attributes were looked through with the commissioner and as all of the needed attributes for the table were already decided during the design phase it was just a case of creating the table. After this the implementation of the student user interface could begin.

In the implementation of the student user interface the first task was to follow the new layout and styles of Asio and to develop the main page for the UI. The layout of the main page and the sub pages were previously decided with the commissioner so the programming could start immediately. The course enrolment packages currently in use in the School of Social and Health Care where used as examples for the functions of the new student user interface. These packages were a great help in the programming of the new user interface and made the progress very fast.

During the implementation phase many problems aroused especially as the author was using new coding techniques like for example the cursor control structures and the highslide element. The commissioner helped a lot also in these cases by providing example cases of the used new techniques. The implementation phase improved the programming and problem solving skills of the author significantly and in the end most problems could be solved independently.

At the end of the implementation phase it was realised that in order for the thesis work to be able to be made public all the names of the used databases, attributes and source code packages needed to be changed for security reasons.

6.4 Testing

Testing of the software was done simultaneously with the development of the software. The testing performed was mainly dynamic browser testing, where the programmed code was executed regularly to test whether particular sections of the developed code where working as needed, and at the same time checked by using a web browser. This was a very efficient way as the errors in the code could be found immediately.

Both Mozilla Firefox 3.6.10 and Internet Explorer 8 were used to test the functionality of the software and possible compatibility issues with different web browsers. At the end of the development phase some test cases were also done where all of the functions of the software were tested from the user’s point of view. The authors own user id was used in the testing as well as a test user id.

6.5 Maintenance and deployment

The new student user interface will be taken into test use by the School of Business and Information management and therefore it will be deployed into the server of Oulu University of Applied Sciences. The test use will most likely bring up some problem cases and software maintenance will be needed. This will be done by the information management department.

If the test use of the user interface proves to be successful the system will be taken into actual use in the course enrolment process of the university and therefore any future maintenance of the system will be done by the information management department.

7 CONCLUSION

Modelling the current course enrolment process of the university exposed the underlying problems and unnecessary actions within the process. These problems were listed and a solution was introduced in the form of a database table for the management of course information and a new student user interface (UI) for course enrolment. The database table together with the new student UI improve the current process significantly by reducing the actions within the course enrolment process and by solving most of the occurring problems.

As well as by reducing the actions needed from the students, the enrolment process has also been simplified by making the usability of the new student user interface as high as possible. As the scope of the thesis work was initially too broad the implementation of the user interface for the database table was excluded from the thesis work. However in order for the current course enrolment process to be as efficient as possible the user interface for the table needs to be implemented.

When modelling the current course enrolment process a lot of the problems of the process were discovered and listed. The ones most relevant for this thesis work were; the managing of the course information, strict selection options for courses which can be set for enrolment, hard coding into the source code, different date notations used, enrolment for all courses and the two separate user interfaces for the students. By implementing a new database table to improve the management of course information and by creating a new student user interface for the course enrolment all of the listed problems were solved.

Even though the parts of the course enrolment process which were addressed in this thesis work have been significantly improved, a lot of work is still needed in order to improve the course enrolment process as a whole. The operation plan course listing done by the study office is very time consuming and should be improved as well as the option to create listings of those students who have not enrolled to any courses. Overall the author believes that the parts of the process concerning the study office should be improved.

Oulu University of Applied Sciences is in the process of improving the quality, compatibility and usability of information within the university's operational processes by taking part in the Raketti project. The aim of the project is to identify the future needs for the study administration and the common processes, which can be supported with information management. The course enrolment process of Oulu University of Applied Sciences had not been previously modelled and no current description of the process exist. Therefore the modelling done in this thesis work can be fully utilised in the OPI subproject when modelling and analysing different study administration processes of Oulu University of Applied Sciences.

8 DISCUSSION

I was very motivated to begin the thesis work and the first phase of the thesis process started fast. Mainly the initiating phase of the process was spent in gathering resources to back up the theoretical background for the thesis work. At the beginning the thesis topic felt difficult and too broad but as the process went on the thesis work was more defined and limited to an appropriate work amount. A risk list was also created at the beginning phase and none of the risks realized during the process. The risks, their effects and preventive actions can be seen in Appendix 2.

The requirements and analysis phase of the thesis process was very demanding as the current process of course enrolment needed to be modelled in order to find out the problems in it. After which the requirements for the new system were defined. The modelling of the current process proved out to be more challenging than expected. As I only knew the enrolment process from the student point of view it was difficult to form a clear picture of what happens behind the scenes. I interviewed both the study office and the information management department to understand the different actions and actors within the process as well as the problems of the process. During this phase the scope of the thesis was also redefined.

Modelling the database used in the process was very time consuming as I had to read through all the source codes of different PL/SQL packages of the system in order to find out which database tables were used in it. After I did the list of all the different database tables, I needed to find out what they were used for. Although time consuming the modelling phase really helped in realizing the requirements for the new system. During this phase I spent a lot of time reading and writing the theoretical background for the thesis.

Implementation and testing phase started behind schedule as I had to work elsewhere during the summer. It was difficult to start the programming of the thesis work because I had not used the software or the programming language in a long time. It took me a while to get used to the software again and I spent some time studying existing source code packages. After the initial difficulties the programming was done quite fast as I was able to use the course enrolment packages in use in the School of Social and Health Care as an example for the new student user interface. Even though the requirements for the new system were clear I only really realized what was required after I had started the programming of the new student user interface.

As I was using cursor structures and an iframe element in the source code, which I had never used before, at first there seemed to be a lot of problems in the source code. After many trial and errors I managed to solve the problems and as time passed my programming skills also improved. I was doing browser testing at the same time which meant that I could immediately see if there were any mistakes in the source code. This made the programming easier and the mistakes in the source code could be easily traced. Toad for Oracle 9.1. tool was very easy to use as in case of any errors in the source code it shows an error code which can be identified in the internet.

As mentioned before the user interface for the new database table needs to be implemented in order for the course enrolment process to be improved and all of its problems solved. Although with the use of table *tame* and the new student user interface the course enrolment process has been improved and most of its current problems solved, the user interface for the new database table is still an important change to be made in order to improve the process. The user interface would not only help the study office when they are setting the courses for enrolment but would also be beneficial for the information management department which would no longer be so involved in the process. In the future with the help of the user interface any type of course can be offered to even a specific study group of the school. If the test run of the new student user interface proves to be successful the user interface could be taken into use in all of the departments of the university.

I learned very much about process modelling during the thesis process and also my programming skills improved significantly. It was interesting and also a bit challenging to do the actual implementation knowing that if the software is good it will be taken into use in the university. The new student user interface for course enrolment will be only taken into test use by the school of Business and Information Management after the thesis process has finished which is a shame as I would have wanted to be able to finish the implementation phase of the thesis work earlier in order to be able to solve any problems which might occur when the test use has begun. Overall I am very satisfied with my thesis work and I was lucky to have a great tutor teacher to guide me through the thesis process and a very supportive commissioner for the work.

REFERENCES

Printed sources

Acuña, S. T. & Ferrè, X. No year of publication. Software process modelling. Argentina: Departamento de Informática, Universidad Nacional de Santiago del Estero. Spain: Facultad de Informática, Universidad Politécnica de Madrid.

Acuña, S. T. & Sanchez-Segura, M. I. 2006, preface. Published in Acuña, S. T. (ed) & Sanchez-Segura, M. I. (ed). New Trends in Software Process Modelling. Series on software engineering and knowledge engineering. River Edge, NJ, USA: World Scientific.

Arlow, J. & Neustadt, I. 2002. UML and the unified process. Practical object-oriented analysis & design. Great Britain: Pearson Education limited.

Eriksson, H. & Penker, P. 1998. UML Toolkit. USA: John Wiley & Sons, Inc.

Issi, L. & Cohen, J. 2004. Web Programmer's Desk Reference: A Complete Cross-Reference to HTML. CA, USA: No Starch Press, Incorporated.

Juran, J. & Godfrey, A. 2001. Process Management. Berkeley, CA, USA: Osborne/McGraw-Hill,

Laguna, M. & Marklund, J. 2004. Business process modelling, simulation and design. Upper Saddle River, NJ: Pearson/Prentice Hall.

Lauesen, S. 2005. User interface design a software engineering perspective. England: Pearson Education limited.

McLaughlin, M. 2008. Oracle Database 11g PL/SQL Programming. USA: McGraw-Hill.

Pohjonen, R. 2002. Tietojärjestelmien kehittäminen. Finland: Docendo Finland Oy.

Pressman, R. 2005. Software engineering: a practitioner’s approach. Boston: McGraw-Hill.

Rob, P., Coronel, C. & Crockett, K. 2008. Database Systems. Design, Implementation & Management. London: Course Technology / Cengage Learning.

Discussions

De Bruijn, O., Planning Officer, Oulu University of Applied Sciences. 2010. Discussion 14.1.2010, 11.2.2010, 14.4.2010, 6.5.2010.

Hedemäki, I., Planning Officer, Oulu University of Applied Sciences. 2010. Discussion 17.3.2010.

Koivukoski, P., Head of Information Management department, Oulu University of Applied Sciences. 2010. Discussion 14.1.2010, 11.2.2010, 14.4.2010, 6.5.2010.

Malinen, S., Senior Planning Officer, Oulu University of Applied Sciences. 2010. Discussion 14.1.2010, 11.2.2010, 14.4.2010, 12.5.2010, 6.5.2010.

E-mail

De Bruijn, O., Planning Officer, Oulu University of Applied Sciences. Re: kysymyksiä, kysymyksiä. E-mail message 18.3.2010.

Malinen, S., Senior Planning Officer, Oulu University of Applied Sciences. Re: kysymyksiä, kysymyksiä. E-mail message 4.3.2010, 18.3.2010.

Electronic sources

Asio-Data Oy. Asio-opiskelijahallinto-ohjelmiston sisältö. Date of retrieval 10.2.2010, http://www.asio.fi/amk/img2.html.

Highslide JS. What is highslide JS. Date of retrieval 15.10.2010, http://highslide.com.

Oulu University of Applied Sciences. What is ehops. Date of retrieval 24.2.2010, http://www.oamk.fi/english/degree_students/career_planning/what_is_ehops.

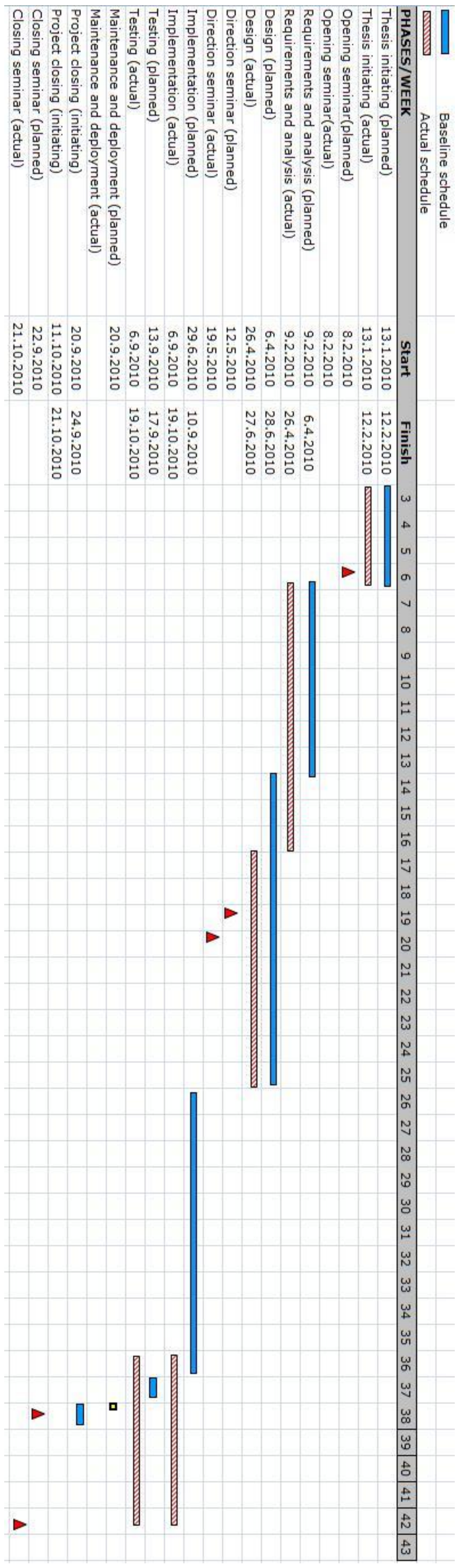
Raketti, rakenteellisen kehittämisen tukena tietohallinto. Date of retrieval 31.3.2010, http://raketti.csc.fi.

Raketti, rakenteellisen kehittämisen tukena tietohallinto. Opi-tavoitemuistio_1.0. Date of retrieval 31.3.2010, http://raketti.csc.fi/opi/dokumentit.

APPENDICES

PLANNED AND ACTUAL SCHEDULE

APPENDIX 1

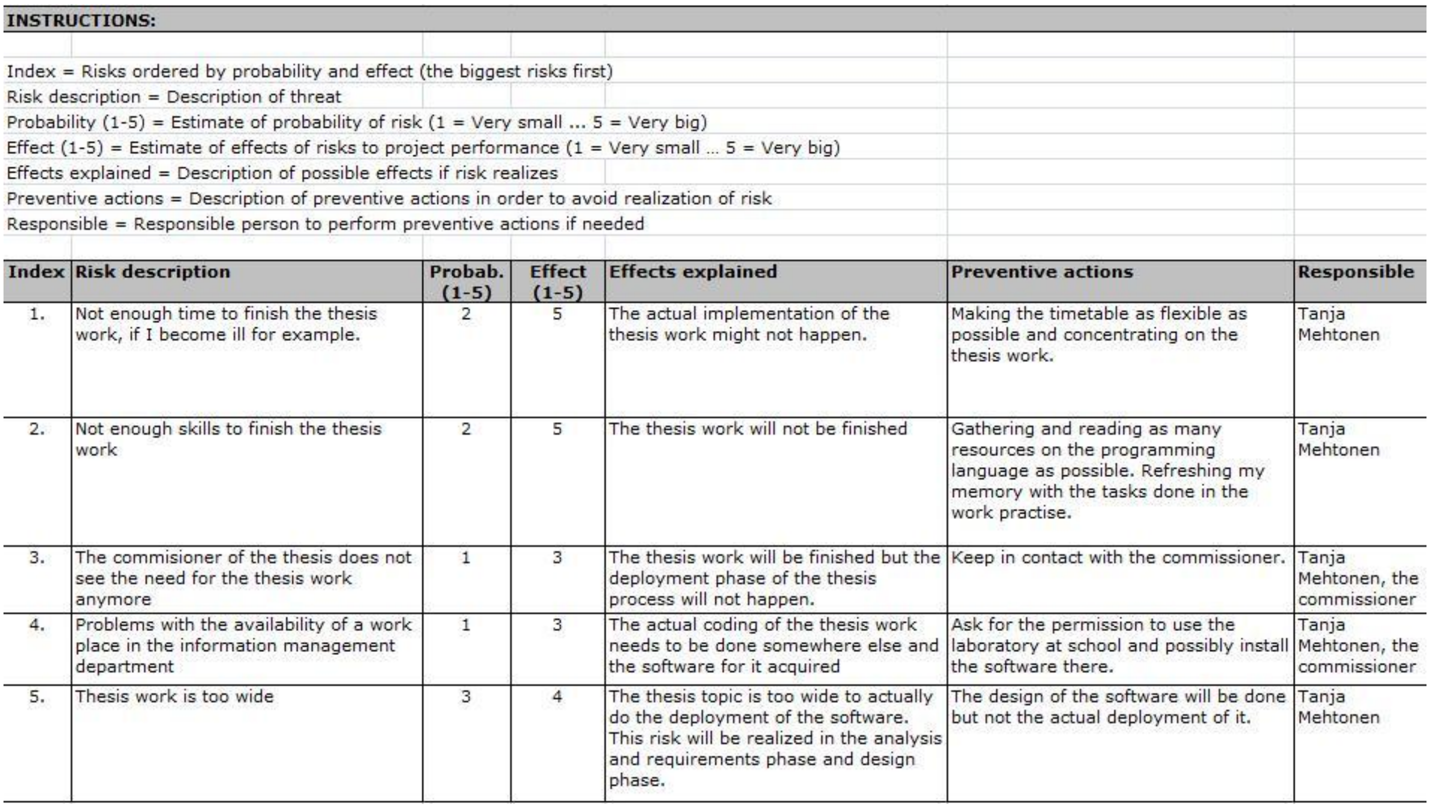


▬ Baseline schedule

▨ Actual schedule

PHASES/WEEK	Start	Finish
Thesis initiating (planned)	13.1.2010	12.2.2010
Thesis initiating (actual)	13.1.2010	12.2.2010
Opening seminar(planned)	8.2.2010	
Opening seminar(actual)	8.2.2010	
Requirements and analysis (planned)	9.2.2010	6.4.2010
Requirements and analysis (actual)	9.2.2010	26.4.2010
Design (planned)	6.4.2010	28.6.2010
Design (actual)	26.4.2010	27.6.2010
Direction seminar (planned)	12.5.2010	
Direction seminar (actual)	19.5.2010	
Implementation (planned)	29.6.2010	10.9.2010
Implementation (actual)	6.9.2010	19.10.2010
Testing (planned)	13.9.2010	17.9.2010
Testing (actual)	6.9.2010	19.10.2010
Maintenance and deployment (planned)	20.9.2010	
Maintenance and deployment (actual)		
Project closing (initiating)	20.9.2010	24.9.2010
Project closing (initiating)	11.10.2010	21.10.2010
Closing seminar (planned)	22.9.2010	
Closing seminar (actual)	21.10.2010	

PRELIMINARY RISK LIST



INSTRUCTIONS:

Index = Risks ordered by probability and effect (the biggest risks first)
Risk description = Description of threat
Probability (1-5) = Estimate of probability of risk (1 = Very small ... 5 = Very big)
Effect (1-5) = Estimate of effects of risks to project performance (1 = Very small ... 5 = Very big)
Effects explained = Description of possible effects if risk realizes
Preventive actions = Description of preventive actions in order to avoid realization of risk
Responsible = Responsible person to perform preventive actions if needed

Index	Risk description	Probab. (1-5)	Effect (1-5)	Effects explained	Preventive actions	Responsible
1.	Not enough time to finish the thesis work, if I become ill for example.	2	5	The actual implementation of the thesis work might not happen.	Making the timetable as flexible as possible and concentrating on the thesis work.	Tanja Mehtonen
2.	Not enough skills to finish the thesis work	2	5	The thesis work will not be finished	Gathering and reading as many resources on the programming language as possible. Refreshing my memory with the tasks done in the work practise.	Tanja Mehtonen
3.	The commisioner of the thesis does not see the need for the thesis work anymore	1	3	The thesis work will be finished but the deployment phase of the thesis process will not happen.	Keep in contact with the commissioner.	Tanja Mehtonen, the commissioner
4.	Problems with the availability of a work place in the information management department	1	3	The actual coding of the thesis work needs to be done somewhere else and the software for it acquired	Ask for the permission to use the laboratory at school and possibly install the software there.	Tanja Mehtonen, the commissioner
5.	Thesis work is too wide	3	4	The thesis topic is too wide to actually do the deployment of the software. This risk will be realized in the analysis and requirements phase and design phase.	The design of the software will be done but not the actual deployment of it.	Tanja Mehtonen

ACTIVITY DIAGRAM OF THE CURRENT PROCESS OF COURSE ENROLMENT

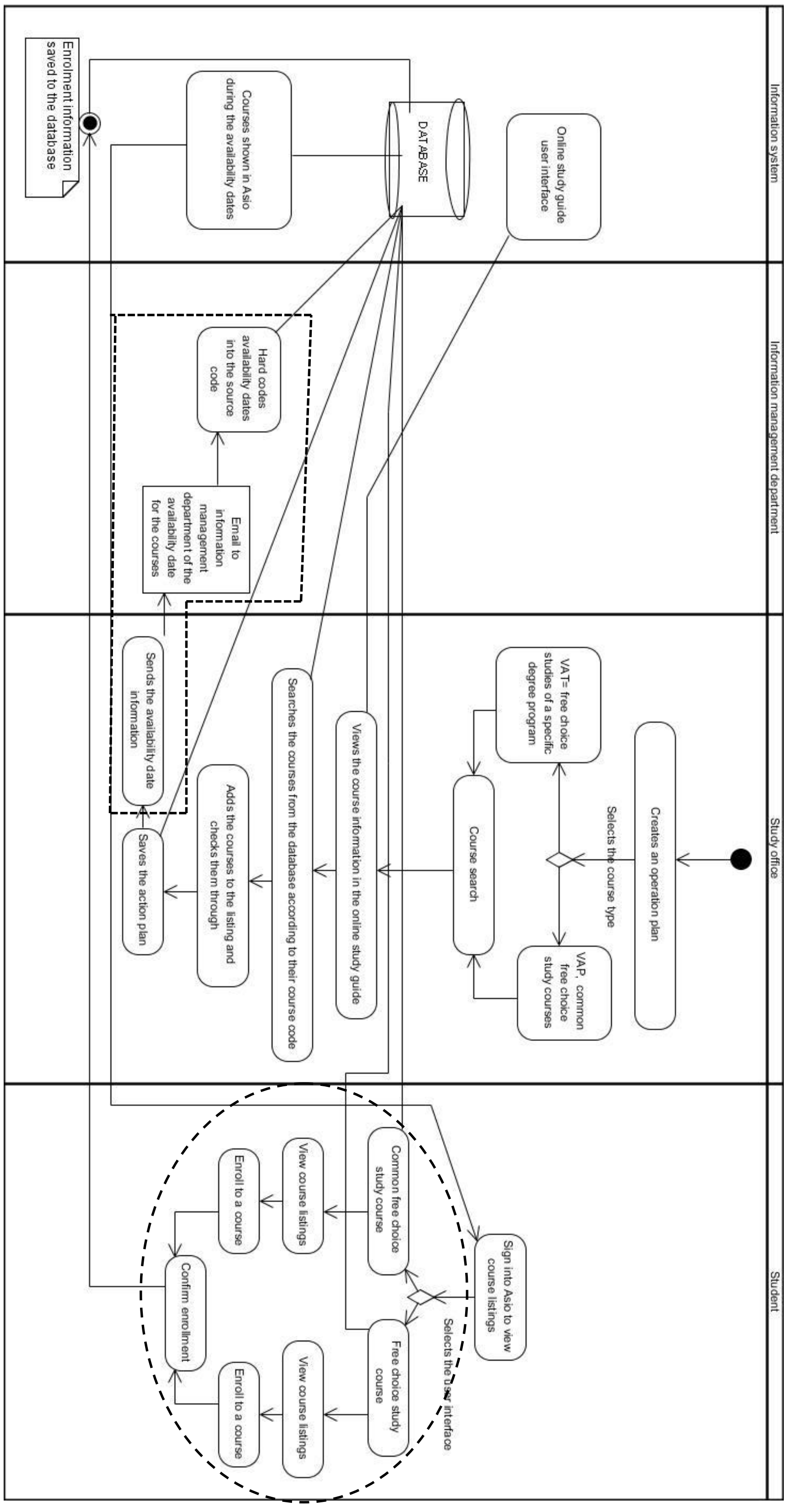

Information system
Information management department
Study office
Student
Enrolment information saved to the database
Courses shown in Asio during the availability dates
DATABASE
Online study guide user interface
Hard codes availability dates into the source code
Email to information management department of the availability date for the courses
Sends the availability date information
Searches the courses from the database according to their course code
Views the course information in the online study guide
VAT= free choice studies of a specific degree program
Creates an operation plan
Selects the course type
Course search
VAP, common free choice study courses
Adds the courses to the listing and checks them through
Saves the action plan
Common free choice study course
View course listings
Enroll to a course
Confirm enrolment
Free choice study course
View course listings
Enroll to a course
Sign into Asio to view course listings
Selects the User interface

TABLE OF CONTENTS