INFORMATYKA EKONOMICZNA BUSINESS INFORMATICS 3(29) • 2013
ISSN 1507-3858

Vangel V. Ajanovski

Saints Cyril and Methodius University, Skopje
e-mail: vangel.ajanovski@finki.ukim.mk

MODEL AND IMPLEMENTATION OF A SELF-ADAPTIVE SOCIAL NAVIGATION SYSTEM FOR PUBLIC INFORMATION SYSTEMS

Abstract: This paper presents a model of a generic navigation system of a public information system, that can be used to improve the structure and content of the information repository via self-organization capabilities based on social navigation and interaction. This model has the primary goal of establishing a generic and adaptive social-based self-structuring navigation system. The model integrates the concepts of social navigation, interaction and self-adaptivity in a feedback control loop. The model focuses on self-adaptivity and includes elements of social navigation in all parts of the system, which enables the implementations based on this model to get social adaptability based on user actions individually, but also as a social environment, in every possible aspect of the functioning of the system. The introduced feedback control loop gives the possibility for further autonomous improvements of the organization of the information. As a proof of concept, this model is then used to build a prototype implementation solution that can be used to guide students towards better course selection during the semester enrolment process.

Keywords: course management systems, curricula recommendations, social navigation.

1. Introduction

This paper proposes a model of a navigation system in an information system, that can be used in public knowledge-bases, information portals, news sites, self-support sites, online directories, etc. The model builds towards enabling the improvement of navigation through the structure and content of a repository of information. In this model, the concepts of social navigation, interaction and self-adaptivity are integrated to enable the semi-autonomous restructuring of the repository navigation.

The original idea was to use a generic approach, and model a system usable in a wide domain of applications. In order to do that, some of the concepts were explored from the very beginnings.

The concept of social navigation is regarded as first defined in [Dourish, Chalmers 1994], and the notion that social navigation can be represented at the following of a path and interacting with other agents (users and content) at certain points in the path, that could lead to changes in the path, as discussed in [Forsberg 1998].

The manner of self-adaptivity that was to be employed is best described in the following snippet from [Laddaga, Robertson, Shrobe 2003, pp. 282-283]: "Self-adaptive software is software that monitors its own operations, detect faults and opportunities, and repairs or improves itself in response to faults and changes. It effects the improvement by modifying or re-synthesizing its programs and subsystems, using a feedback control-system like behaviour". So, based on such generic concepts, two main groups of functionalities that are expected from the new navigation system model were set as requirements:

- the possibility to record social interaction at all points in the system, either between the system and the visitor or among visitors that are (concurrent or not) visiting the same place of interest in the system – sharing, pointing, recommending and monitoring the published resources and paths;
- use the social-interaction history to improve the organization of the navigation and the structure of content – autonomous self-change of the navigation elements.

2. Navigation system model

In order to include the requested functionalities, the model first provides the ability to change all navigation point and paths, the whole structure, and then enables social interaction that is associated to the relevant versions of the elements, current to the moment that the interaction has happened. In addition to this, the model enables controlled self-adaptivity, according to the results of the performed analysis based on social interaction history.

The first step in building this model is the separation of the navigation structure from the content by using a separate navigation sub-system. This sub-system should use a structure that defines (on a conceptual level) and lists (on a logical level), all the basic navigational elements that are interesting from a social point of view, and should also include a structure that performs the mapping of logical elements URLs. This separation would allow the independence of the physical/logical level – important points for the realization of the structural adaptivity of the system because the constant change of the navigation elements can be the enemy of social navigation (in terms of sharing changed URL, broken paths etc.) The importance of such aspects is discussed in [Dieberger 1997].

For the same reasons, the navigational structure must allow the storage of the entire history of changes (versions of navigation elements) and to handle the shifting of resources in a controlled way that will not disrupt the navigation.

2.1. Navigation elements

The base model is organized in three levels. At a conceptual level, we differ between the following basic types of social navigation elements (which can be nested in order to achieve complex structures):

- atomic resource is the basic element (e.g. a resource of its kind is the “Description of the course Databases in CS study program”, but can also be external resources and applications), which can exist only as part of an
- unordered set of logical resources (example: “Guidelines for undergraduate studies” or “Materials for Learning Databases”), or an
- ordered set of resources or a directed path (example: “Installation Guide to Oracle DBMS”).

At a logical level, a logical navigational structure of the overall system is built and individual resources or resource sets and their connections are enumerated.

At a physical level, the mapping between atomic resources and physical resources, or addresses, is performed.

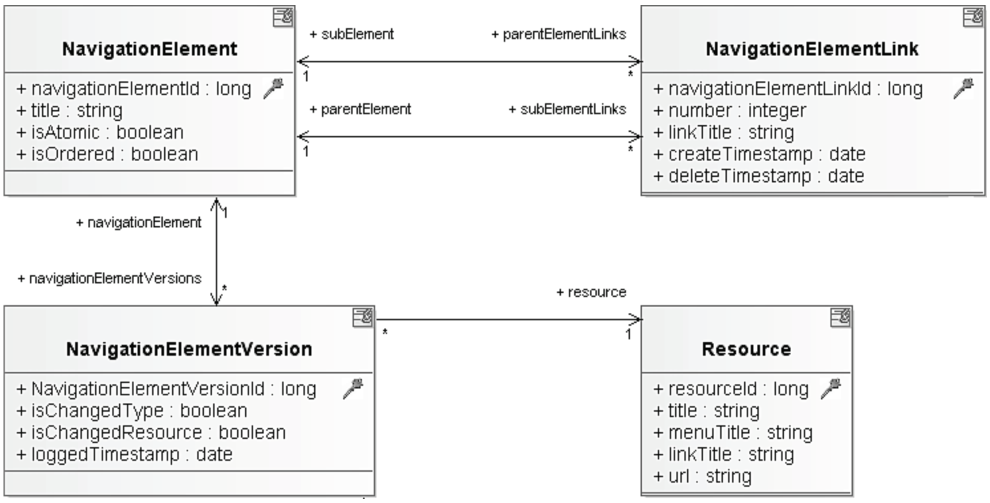


Figure 1. Navigation model elements

Source: own elaboration.

Figure 1 shows the model of the navigation structure. The class NavigationElement represents all of the navigational elements that define the logical conceptual level. The type of each element is indicated by the attributes isAtomic and isOrdered. Logical links between navigation elements are defined with the NavigationElementLink class. The Resource class provides the physical mapping of various navigational elements to specific addresses (URL) in the WWW space.

2.2. Supporting self-restructuring in the model

The presented basic conceptual model is then extended to allow self-restructuring. For this purpose, the system should support the changeability of the structure and the changeability of resources. This means that one can change the order of resources (replacing one type to another, adding and deleting logical resources) and can change the final physical content resources.

Changing the content and resources

By changeability of resources, we actually mean replacing the elements at the physical level. The intention is to only change the mapping of logical resources to physical addresses. Also the history of all changes to the mappings should be kept in order to enable later analysis of the behaviour of the users related to each change, and pin-point the appropriate version of a resource that is of interest.

The model further elaborates version keeping. In the second row of classes in Figure 1 the versions of the elements and all changes to the physical level are presented. The mapping of physical resources is such that a navigational element is not directly associated with a single physical resource, but in fact a new version of the navigation element is created for any change in the physical mapping. Each change creates a new version object which records that the version is associated with a change of some resource to a new address. The date of the change is also kept.

In this way the version entries of each mapping can introduce new versions of the same physical resource – on another date, which would indicate that the navigation element references the same source, again and again, but on different dates. This can be interpreted as if the content of the source is changed from time to time, at the version dates, but the URL is still the same. Also, a completely new version can be created for a new physical resource with a new address. The current version is considered to be the one with the latest time-stamp.

Changing the navigation structure

The structure of the navigation system can be changed by modifications of the set of navigation links. But, in order to effectively monitor the changes of the structure and the continuity of social navigation – i.e. enabling continuous monitoring of which were original interests of users and where they have migrated after a change, this model only allows for elementary changes and not a fundamental change-set where all traces of the past would be gone. The idea is to maintain the traceability of each change of the structure.

Only several basic operations are allowed that are implemented in a way that allows the monitoring of changes:

- change the type of a logical resource;
- adding new resources to a logical set;
- removing a resource from a logical set;
- moving a resource from one to another set.

Considering the way how the logical resources are implemented in the data model using navigational elements – then the allowed set of change operations can in fact completely rearrange the navigation structure in any form when performed as a composition. So in fact any modification can be made but it must be performed gradually, in order to keep proper records of associations of the navigation elements and their versions with the social interactions.

Changing the types of resources

Swapping a navigation element with another kind is permitted for ordered and unordered sets. As an example, instead of an unordered set of resources, we can decide to have a directed path through the same set of resources or vice versa.

This change is easy to implement over the existing class versions of navigation elements (see Figure 1), with the added ability to record whether a new version includes change of type. At the same moment, a change of the resource can also be indicated. It is done in this way because it can often be required to reference a new page for the new type and it would be unnatural if, even for a small amount of time, first a change of type is visible and immediately after, a change to the mapping to a new content. A change of the element type that does not require a change to the physical resource will be created as a new version and the relevant attributes will be denoted, but this new version will still reference the (old) resource. This is done in order to obtain better performance since search operations are many times more frequent than structural changes.

Adding a logical resource to a set. This operation is realized by creating a new navigation element (if it is a new resource) and then adding a link of the element to the set. Each link keeps a record of the date of creation, which allows one to find out which links were available in a certain period of time.

Note: If the set was made up of only one element, that is atomic, the type of set (isOrdered) has no meaning. So, it is ambiguous what will happen once a new element is added, unless first the administrator manually decides on the ordering.

Removal of a logical resource. This operation is realized by setting the value of the attribute deleteTimeStamp in the link that connects the element (child item) to the resource set (parent item). Thus historical data needed for decision making and analytics will not be lost, and navigational elements themselves will continue to exist in the system in case a new reference to them is needed in another place of the navigation structure. Elements that are no longer connected to other elements are considered orphan elements and can be monitored in case they are needed again in the future.

Note: the re-inclusion of a navigational element that has already been connected in the past, does not change anything in the historical records, but simply creates a new link by performing the operation of adding an existing resource set.

Moving a resource from one to another set. This operation is the same as the composition of two operations: remove a resource from the old location set, and add the existing resource to the new location set.

2.3. Social navigation part of the model

In order to enable basic support for social navigation throughout the whole of the navigation system, it is required to keep records of all interactions that have occurred, for each interaction type, with each type of navigational element, and to keep additional parameters for each interaction that occurred. Interaction (viewed, read, accessed) should be recorded for the current version of a navigation element and for the link between elements that was interacted with (followed, clicked, etc.).

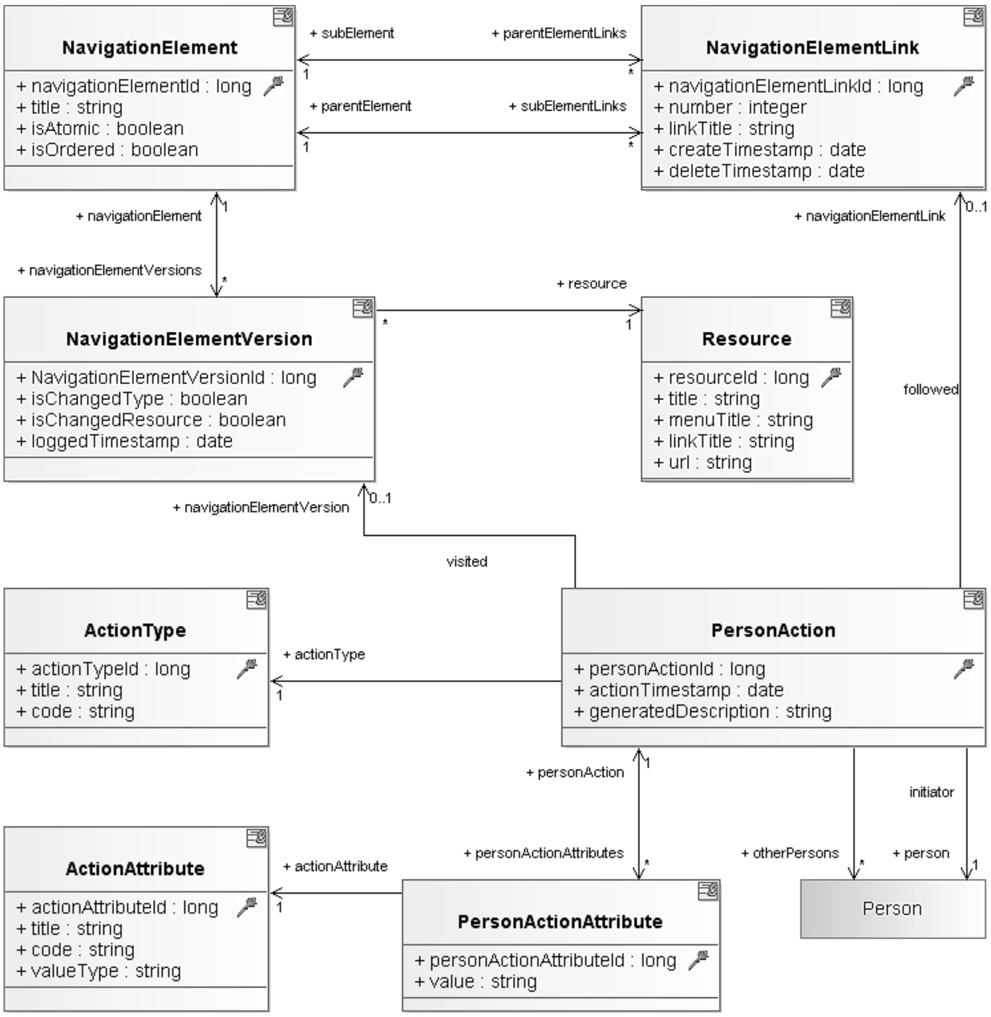


Figure 2. Data model of social navigation features

Source: own elaboration.

These requirements are realized through an extension on the previous model, presented in Figure 2. All types of actions that a person can make (indicated by the Person class) are codified according to ActionType and are kept in a log of taken actions (class PersonAction), sorted by the action time stamp.

The actions that have a special meaning and are specially recorded in the journal are:

- interaction with other visitors (list of persons to whom the communication is directed);
- interaction with a navigational element (actual version of the element);
- interaction with a navigational link between two elements.

We have also defined attributes that can be of interest to monitor in association to the actions of the visitors. Such attributes and their values can be recorded. What is more, there is no strict list of attributes, but this is left free to define depending on the needs.

The Person class is not specified in detail, but should be further elaborated in the implementation of the model. In the case of public information systems where the majority of visitors are people that we are not familiar with, there are two options for addressing these visits:

- Anonymous option – to use only one anonymous object: Person, toward which all actions are recorded.
- Weak authentication – for Each new visitor the system creates a new object of the class Person unless there are kept data from a previous visit, such as data kept in cookies or some other technique to determine the ID of a visitor that is actually returning to the system.

2.4. Self-adaptivity part of the model

The previous discussion identified many elements that allow a constantly changing navigation structure, so it is necessary to monitor all the changes that have occurred. This is necessary for various reasons, the most important being quality control and the ability of the system to constantly adapt to the new needs.

Background on self-adaptivity

The analysis of the literature in the field of software engineering self-adaptive systems showed different aspects that need to be addressed in such systems, most of which were not relevant for the purposes of social navigation.

The model shown in Figure 3 represents a feedback control loop in a self-adaptive system [Dobson et al. 2006]. This model was originally discussed for communication systems, but can be applied in terms of software systems [Lemos et al. 2011].

This model of self-adaptive control can be viewed as an appropriate match to the navigation model proposed in [Spence 1999] (see Figure 4) and the amendments for social navigation discussed by [Riedl 2001].

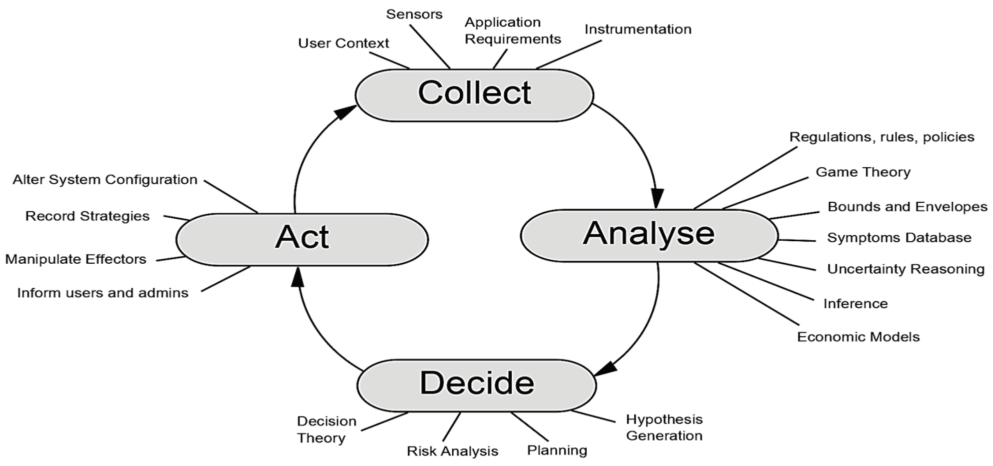


Figure 3. Feedback control loop

Source: own elaboration.

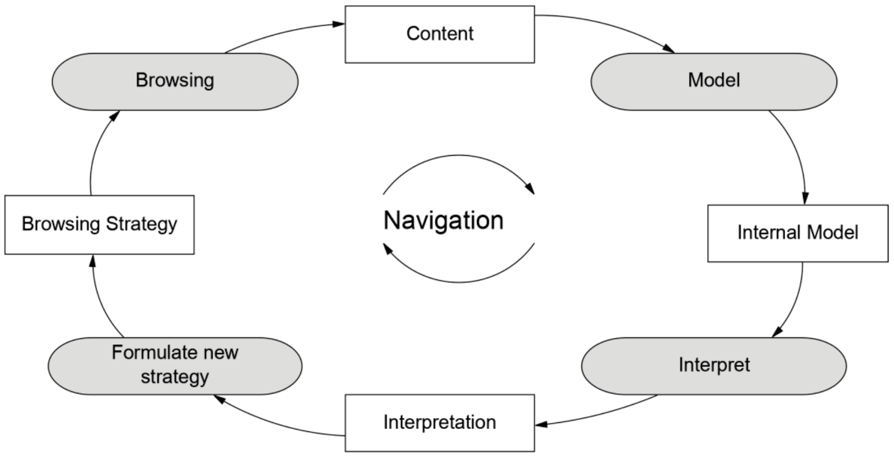


Figure 4. Navigation model by Spence

Source: own elaboration.

Table 1 shows the alignment of the concepts of both models. It is almost obvious that one can establish a relation between the two discussed models. The new model uses a feedback control loop to enable self-adaptivity for social navigation in the system.

These are two concepts that are completely different in nature; one explains the cognitive process of navigation from the viewpoint of the visitor as a navigator, and the other describes the process of controlling the operation of a software system that alters its own parameters. The idea of linking these two concepts lies in the need for

Table 1. Cycles of social navigation and self-adaptivity

Social Navigation	Self-adaptivity
Browse	Collect
Model	Analyse
Interpret	Decide
Formulate New Strategy	Act

Source: own elaboration.

a self-adaptive feedback control loop in order to improve the social navigation over the entire set of visitors. The mapping of the cycles in this context would mean:

- browsing information through the system by visitors generates large amounts of navigational data that characterize the overall behaviour of the system, and should be analysed;
- symptoms should be monitored and controlled by the navigation system, as they can have an impact in the formation of a wrong cognitive model of the navigation structure with visitors and as a response, the visitors could abandon the system entirely;
- visitors build their own cognitive model based on the presentation of the structure and on previous experience, so one cannot directly affect them immediately – but an analysis of user behaviour can be made, in order to provoke responses in future cycles;
- the strategy that is formulated can be affected by even the simplest actions of the system and indicators put on well-chosen positions, thus it is crucial to monitor changes through cycles for the evaluation of success of such elements.

All these features are implementation dependent, hence cannot be part of a generic model. What can be generalized is the execution of the control cycles and their mapping into a generic data model, which will provide the necessary analysis regardless of the implementation at hand.

Model structure

In social navigation systems it is necessary to monitor all activities of visitors and their usage of resource in order to establish rules of behaviour. This requires a detailed data model with the following features:

- track all actions that occur within a cycle of the self-adaptive control loop;
- track the evolution of all system and process parameters across all cycles of the system.

A data model is proposed first, only for the control cycle and it is shown in Figure 5. This model enables the monitoring of the system through all the cycles. After that this model is linked to the data model for social navigation.

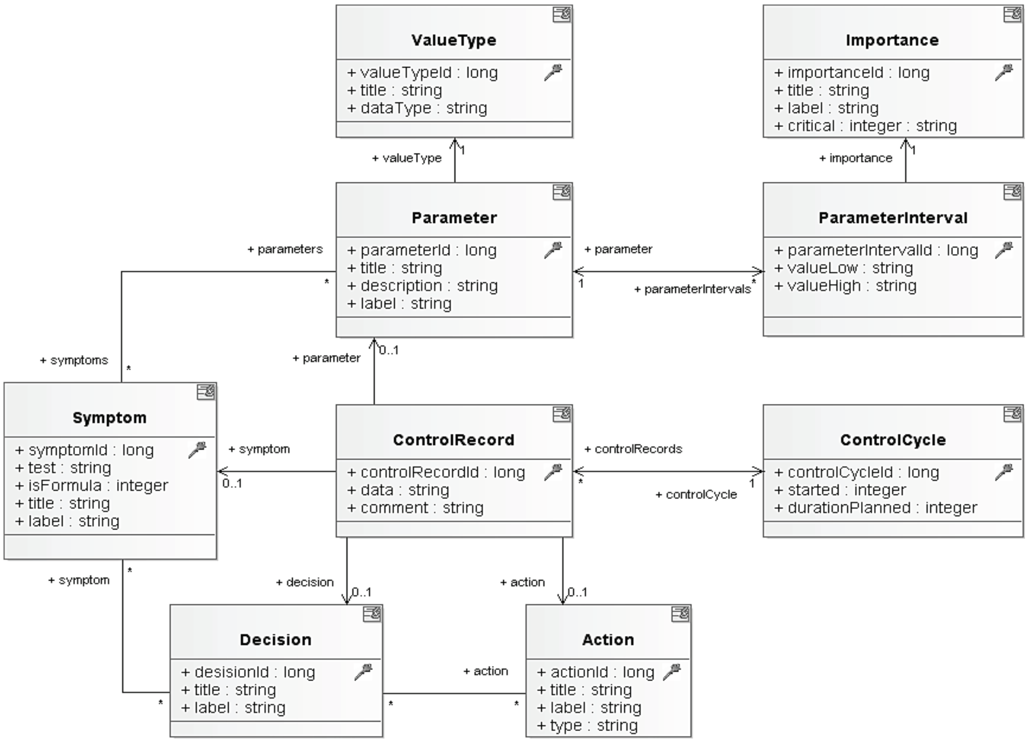


Figure 5. Data model of the control loop

Source: own elaboration.

The part of the model that defined the feedback control loop is used to store data about:

- Sets of defined values of the measured systemic and procedural parameters, normal range of values, increased boundaries and critical limits – in order to analyse the status of processes and objects in the system
- Journal of measured values of parameters, identified symptoms, decisions made and actions taken.

The journal is of particular importance because it tracks the success of the control cycle and can be used to find out if the correct symptom was identified, whether the requested action at the time was the one that was really needed, and if the results can be seen and the history of decisions checked and conclusions drawn on the correctness of the decisions. Of course, some testing would be done manually by the analytical team and some analysis should be programmed for automated monitoring to work.

2.5. Integration of the social-navigation and self-adaptivity model parts

In order to follow the partial change of the structure of the system, the impact of each change on the behaviour of users and the impact of changes over the general parameters of the system and boundaries that are allowed, the system records the changes that have occurred in each self-adaptive cycle.

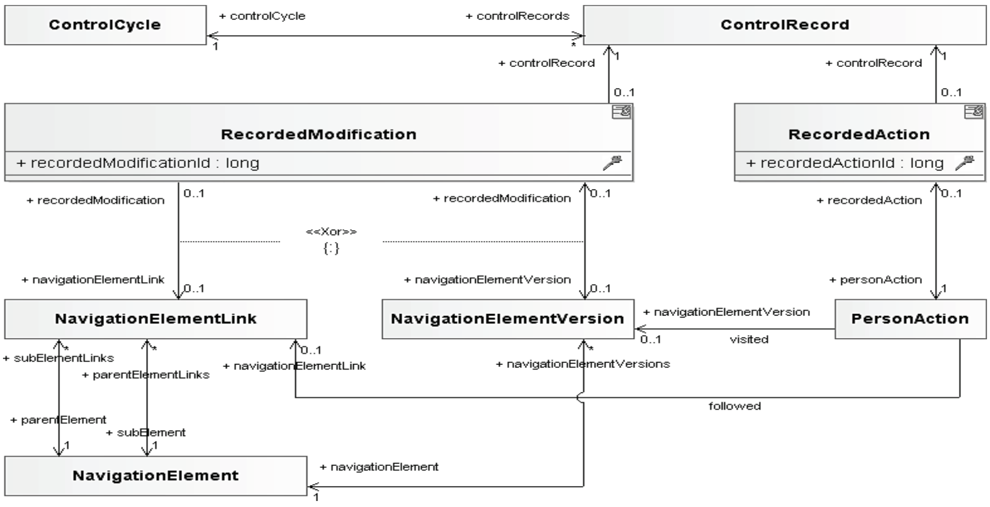


Figure 6. Integration of the models of social navigation and self-adaptivity

Source: own elaboration.

The model that describes these records is shown in Figure 6. It should be noted that all associations are optional, which means that the model allows for the incoherent behaviour of the two segments and allows selecting only those components that are really needed for the implementation. The RecordedAction class is used to store the information that a certain PersonAction took place within a ControlCycle and is associated with a ControlRecord. Similarly, the class RecordedModification is used to store the fact that a modification of the structure of content took place and is related to the ControlRecord.

2.6. General considerations for implementations of the integrated model

The steps to implement this generalized model in a production system include the final consideration of the requirements of the target system and the amendment of the structures' necessary attributes – especially time stamps, recording the users who make administrative changes in the structure and the like.

In cases when a new information system is built from the beginning, it is recommended to use this model in all stages as an initial model for the design of the navigation system.

On the other hand, if the information system has already been built and it is only required to add some of the previously described social and/or self-adaptive functionalities, the first consideration that should be discussed is the feasibility of the complete replacement of the existing navigation system and the development of a new navigation system and all its functionalities as an integrated component of the existing information system. If this is not feasible, the only solution is the implementation of the navigation model as a separate navigation system, and then mapping and forwarding copies of actions from one to the other system that will keep the two systems in sync.

3. Prototype implementation

This integrated model can be applied in many different areas, but primarily set in information systems aimed at presenting knowledge and processes related to the management of knowledge. Example implementations can be:

- Information portals.
- Directories and databases of knowledge system.
- E-learning.
- Social networks oriented learning.

3.1. Background information

As a proof of concept, the presented model was used to implement a social-navigation based virtual academic adviser [Ajanovski 2011], that uses recommendations based on social interaction to guide students in the process of course selection and the creation of a personal study plan until the end of the studies on a longer time scale [Ajanovski 2013c], but also as a self-adaptive control mechanism to create the best possible class schedule in terms of fewer clashes between classes [Ajanovski 2013a].

In this implementation the repository is in fact the catalogue or database of curricula and study plans for all study programs at university level.

The virtual academic adviser gives the student more personal guidance in the process of each semester enrolment, in order to come up to a personal study plan that will be personalized regarding the study context, situation and capabilities. The idea is that the student can analyse this plan, compare it with other students' plans, try various alternatives with changing the order of courses, browse for interesting courses, choosing from the electives, number of credits per semester and even switch to a new study program. Once satisfied, he/she can enrol in the proposed courses for the next semester. If anything is unclear, the student can still go to the real adviser and discuss the future plan together.

The social-navigation features are implemented as part of the browsing experience for courses from the course catalogue, while the self-adaptive features are implemented as part of the background services that control the number of students per group, availability of groups and time-slots, right up to the choosing of the algorithm for course recommendations.

The first version of the virtual academic adviser enabled the student to try a ‘what if’ experiment with the various choices on offer – such as number of credits per year, choose another study program and specialization profile, rearrange the order of enrolment of courses per future terms etc. After these experiments, the student would decide on some preferred scenario and enrol for the term according to the profile and the official rules. In case there were any issues with some scenarios, the student could discuss them with the real adviser. It should be noted that the role of the adviser is to give advice and not decide on behalf of the student, and once the choice is legitimized, the adviser cannot prevent the student from enrolment, but can only suggest better options.

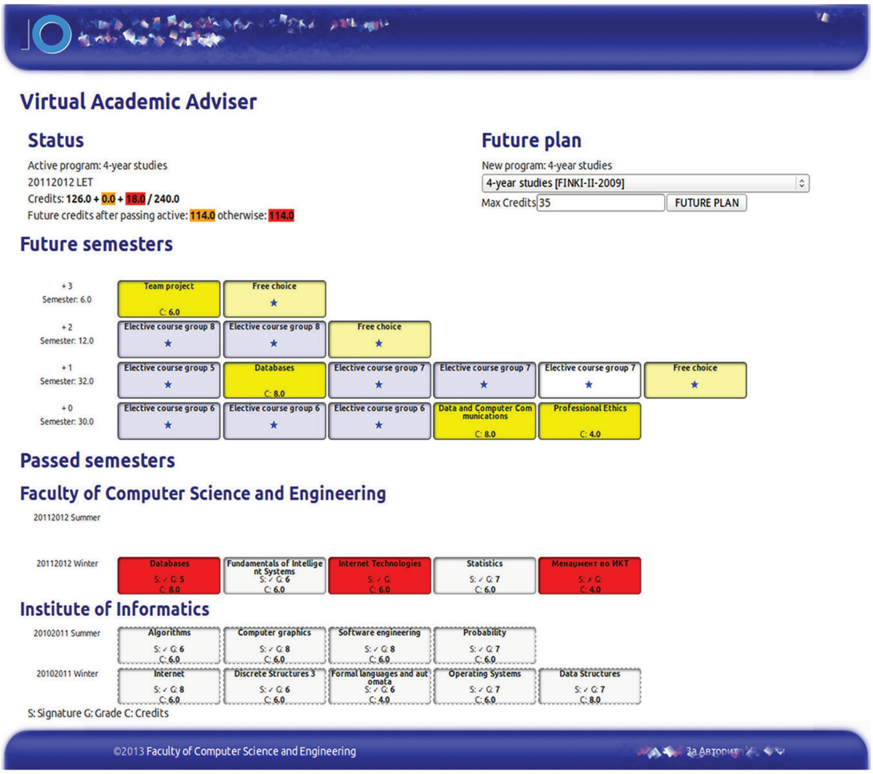
With the implementation of the new model, the second version of the virtual academic adviser is developed with two additional features: giving the students manual and automated course recommendations and the introduction of a new integrated process for term enrolment, class scheduling and construction of timetables. Within this evolution, mechanisms for mutual dynamic self-regulation of the processes of term enrolment and class scheduling are investigated.

Current implementation of the virtual academic adviser

In the current implementation of the virtual academic adviser (see Figure 7), each row represents a semester and each box in the row is a course enrolled in that semester. The semesters are ordered in such a way that the last or active one is on the top, and downwards follow earlier enrolments. Each box shows the name of the course, whether the lecturer has certified the student was present at the majority of lecture hours and is allowed to take the exams, the final grade of the student and how many ECTS credits the course is worth.

The boxes for the past semesters are colour coded so that the green boxes represent active courses in the current semester, the orange boxes represent courses where the lectures have finished but the student did not have a chance to pass yet and the white boxes represent courses that the student passed; the red boxes represent courses that the student failed.

The boxes for the future semesters have a star whenever recommendations for a course exist in a certain slot and they are colour coded as well, so that the yellow boxes represent mandatory courses, the light blue boxes represent elective slots where a choice from a fixed group is possible and the light green boxes represent slots where the student has a completely free choice.



Virtual Academic Adviser

Status

Active program: 4-year studies
20112012 LET
Credits: **126.0** + **0.0** + **18.0** / **240.0**
Future credits after passing active: **114.0** otherwise: **114.0**

Future plan

New program: 4-year studies
4-year studies [FINKI-II-2009]
Max Credits 35 FUTURE PLAN

Future semesters

+ 3 Semester: 6.0	Team project C: 6.0	Free choice ★				
+ 2 Semester: 12.0	Elective course group 8 ★	Elective course group 8 ★	Free choice ★			
+ 1 Semester: 32.0	Elective course group 5 ★	Databases C: 8.0	Elective course group 7 ★	Elective course group 7 ★	Elective course group 7 ★	Free choice ★
+ 0 Semester: 30.0	Elective course group 6 ★	Elective course group 6 ★	Elective course group 6 ★	Data and Computer Com munications C: 8.0	Professional Ethics C: 4.0	

Passed semesters

Faculty of Computer Science and Engineering

20112012 Summer					
20112012 Winter	Databases S: ✓ G: 5 C: 8.0	Fundamentals of Intellige nt Systems S: ✓ G: 6 C: 6.0	Internet Technologies S: ✓ G: C: 6.0	Statistics S: ✓ G: 7 C: 6.0	Менаџмент во ИКТ S: ✗ G: C: 4.0

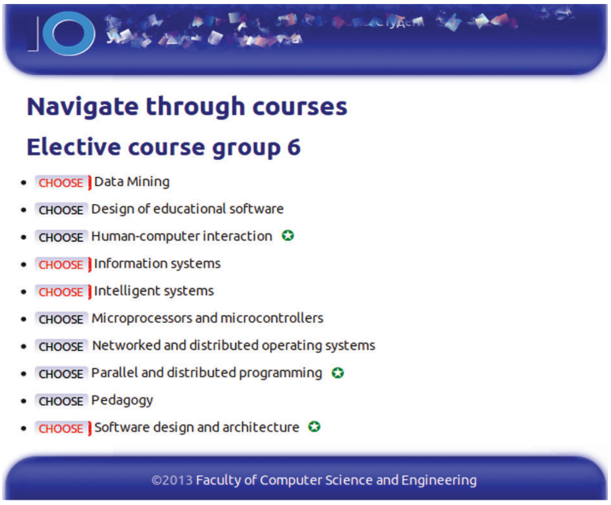
Institute of Informatics

20102011 Summer	Algorithms S: ✓ G: 6 C: 6.0	Computer graphics S: ✓ G: 8 C: 6.0	Software engineering S: ✓ G: 8 C: 6.0	Probability S: ✓ G: 7 C: 6.0	
20102011 Winter	Internet S: ✓ G: 8 C: 6.0	Discrete Structures 3 S: ✓ G: 6 C: 6.0	Formal languages and aut omata S: ✓ G: 6 C: 4.0	Operating Systems S: ✓ G: 7 C: 6.0	Data Structures S: ✓ G: 7 C: 8.0

S: Signature G: Grade C: Credits

©2013 Faculty of Computer Science and Engineering

Figure 7. Screen-shot of the current state of the virtual adviser (anonymized data).



Navigate through courses

Elective course group 6

- CHOOSE Data Mining
- CHOOSE Design of educational software
- CHOOSE Human-computer interaction ✪
- CHOOSE Information systems
- CHOOSE Intelligent systems
- CHOOSE Microprocessors and microcontrollers
- CHOOSE Networked and distributed operating systems
- CHOOSE Parallel and distributed programming ✪
- CHOOSE Pedagogy
- CHOOSE Software design and architecture ✪

©2013 Faculty of Computer Science and Engineering

Figure 8. Choosing elective courses from a group, with recommendations

Source: own elaboration.

Clicking on a slot marked with a star gives the student a list of all relevant courses that he/she can choose in that slot. Depending on the student that accesses the catalogue, some courses are shown with icons that indicate a recommendation – if the course would be beneficial to the student in terms of relevancy to his/her specialization, forecast that he/she might achieve a good grade, most popular courses by his/her peers, etc. (see Figure 8).

The system takes into account all interdependencies and course prerequisites and will propose a *realistic plan*. But whether this plan will succeed, depends only on the ability of the student to follow and keep precisely to the plan and pass all the requisites.

The virtual academic adviser is also mobile-enabled (see [Ajanovski 2013b]), allowing access and usage at any location, with the possibility for the student to save their plans as navigation paths and access them later without waiting for the computation and also enabling the discussion and sharing of the plans with other students (if they wish). This is made possible by the background social-navigation model that was previously discussed.

3.2. Specifics of the implementation type

This implementation is of the mapping type, since the existing system (ISIS) was developed several years ago. The old navigation system was kept, and new navigation system based on the new model was built only for enabling the social navigation of courses, topics and study programs.

Mapping between the previous system content and new system navigation

This is enabled with the introduction of a mapping entity class SocialModelMapping that in fact keeps information on triplets (className, objectId, navigationElementVersion):

- className is the class in the original model that was mapped to a navigation element in the new model;
- objectId is the identified of the object that is mapped from that class;
- navigationElementVersion is the version of the NavigationElement that is mapped to.

Using this class, a mapping is done so that:

- study programs from the old model become top level navigation elements sets;
- courses within them become nested elements;
- slots that are elective are links between a set and navigation elements.

When students browse through the catalogue, a journal is kept of all the links they have taken and all the navigation elements they have visited, in order to be able to perform analysis and find out about trends, interests, similarities, etc. that are used for the recommendations.

Implementation of the self-adaptive model

The control loop was implemented as a component within the existing information system and used to orchestrate the process of enrolment, phase by phase, student by student or group by group whatever is relevant. In the following few paragraphs the four steps of the control loop are explained with all the included mechanisms.

In the **Collect** step, the system is monitored and information needed from the students and other system components will be gathered:

- critical time-slots that are almost full,
- length of student waiting lists,
- numbers of students per status of the enrolment,
- numbers of students per year, per group, per status,
- number of issues reported by students per category,
- number of students without grades for past enrolments.

In the **Analyse** step, the operational status of the system is analysed according to the gathered parameters, historical data and boundary values for several symptoms that are identified in the symptom database:

- new groups will be needed soon on a course,
- new teachers will be needed soon on a course,
- course resources are exhausted,
- courses will not be activated due to lack of students,
- students ask for courses that are not on offer,
- student grades are not input on time,
- increasing numbers of students have complaints.

In the **Decide** step, the system makes decisions on the actions that are to be performed, depending on the severity of the symptoms encountered and how critical the values of the monitored parameters are. In this case, the decisions should be mainly made on when and how to act:

- send only information and status via e-mails,
- invoke critical alarms to administration staff,
- boundary limits can be changed because the number of students expected will not exceed significantly,
- ask advice on action from administration staff.

In the **Act** step, all actions are orchestrated based on the types of decisions that were made and which symptoms were triggered. The actions that are present are:

- status information is sent to all users,
- administration staff is informed that there are issues present together with the symptom,
- analysis, respective numbers and possible decisions for the decision database,
- critical alarms are activated per symptom,
- boundary limits are modified and the action is logged,
- decisions are logged,
- symptoms are logged,
- measured parameter values are logged.

In such a process the course enrolment and time-table creation can be monitored, analysed and acted upon automatically or manually via the proposed control loop framework. In this case it can be argued that the process is successful if finished on time – before the official start of the semester. If that is not the case, the framework gives the possibility to monitor the percentage of finished cases of the student term and course enrolments' and gradually increase the severity of symptoms and the frequency of issued critical notifications to the administration staff and to students that have not been active.

Aligning Components of Social Navigation

The import of historical data from the course enrolment system that is part of ISIS into the recommendation system is completed in a few minutes, and the three core algorithms for standard analyses included in the recommendation system run for an additional four minutes within the development environment.

These data show that the recommendation systems can be used for a dynamic self-test on a daily basis according to the following scenarios for automated experiments and analysis of performance:

- daily loading of the test sample from the previous cycle in a clean database and evaluation of the performance of several algorithms,
- daily selection of new test groups of students who are given recommendations based on different algorithms and tracking the acceptance of the issued recommendations,
- daily comparison of the results to two control groups — students with randomized placebo recommendations and students using a static version of the system without recommendations.

3.3. System architecture

The proposed architecture was conceived as a web-based solution. Only components and sub-systems that are free and open-source were used.

Main Sub-systems

The front (visible) side of the web-based user interface is written using standard HTML code, CSS styles and minimal use of JavaScript. The back-end logic is developed in Java, using the Apache Tapestry (http://tapestry.org) framework for web applications. The mechanisms used to link all the tiers of this application are part of the basic framework used – Tapestry IoC (for DI and IoC).

The implementation of the data model is done with ORM, using on JBoss Hibernate (http://hibernate.org). This library was chosen because of the possibility of mapping to different DBMS-s and due to simplified migration.

PostgreSQL is one of the most popular relational database management systems with an open and free source code (http://postgresql.org). Many years of experience

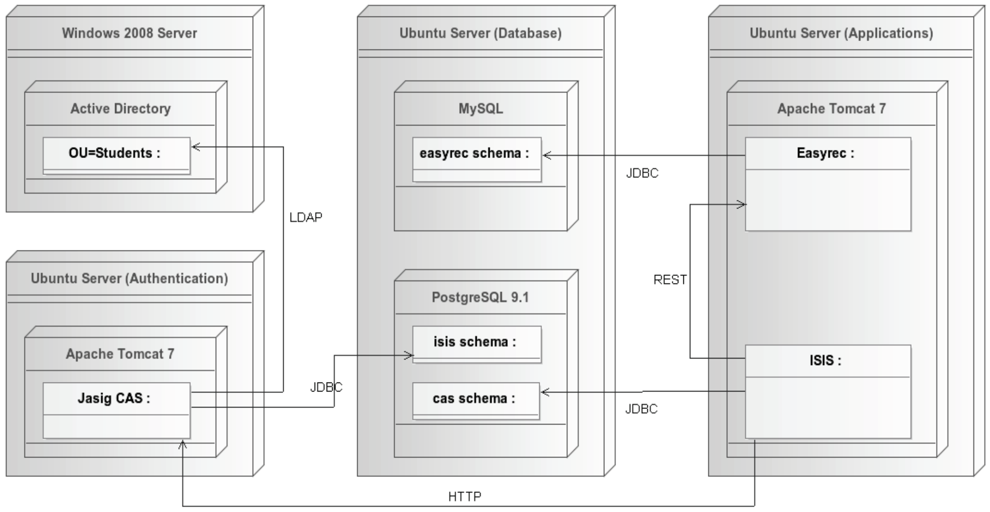


Figure 9. Deployment diagram showing all sub-systems

Source: own elaboration.

using PostgreSQL as a server for RDBMS for the online course management system – Moodle (http://moodle.org), is one of the most important reasons for choosing this system.

After many years of experience within the Faculty, for the authentication system we use Jasig CAS (Central Authentication Service) (http://www.jasig.org/cas). This system is designed for central authentication, and can be used by multiple web applications.

Easyrec is used as a system for recommendations; it is an open source solution and uses a modular architecture that makes it suitable for easy integration with other systems (http://easyrec.org). The basic recommendation engine that is built-in and was used in the experiment is very simple, but the system is easy to extend with additional custom plug-ins and it can swap engines online.

Figure 9 shows the deployment diagram of the main sub-systems and the interconnection between them.

Internal architecture

The applications in the system are organized according to the following prin-ciples, and an internal four-tier architecture is used:

- graphical user interface written in HTML,
- background processing for the interaction and logic of each web-page, written in Java,
- service classes for inter-connections and business logic, written in Java,

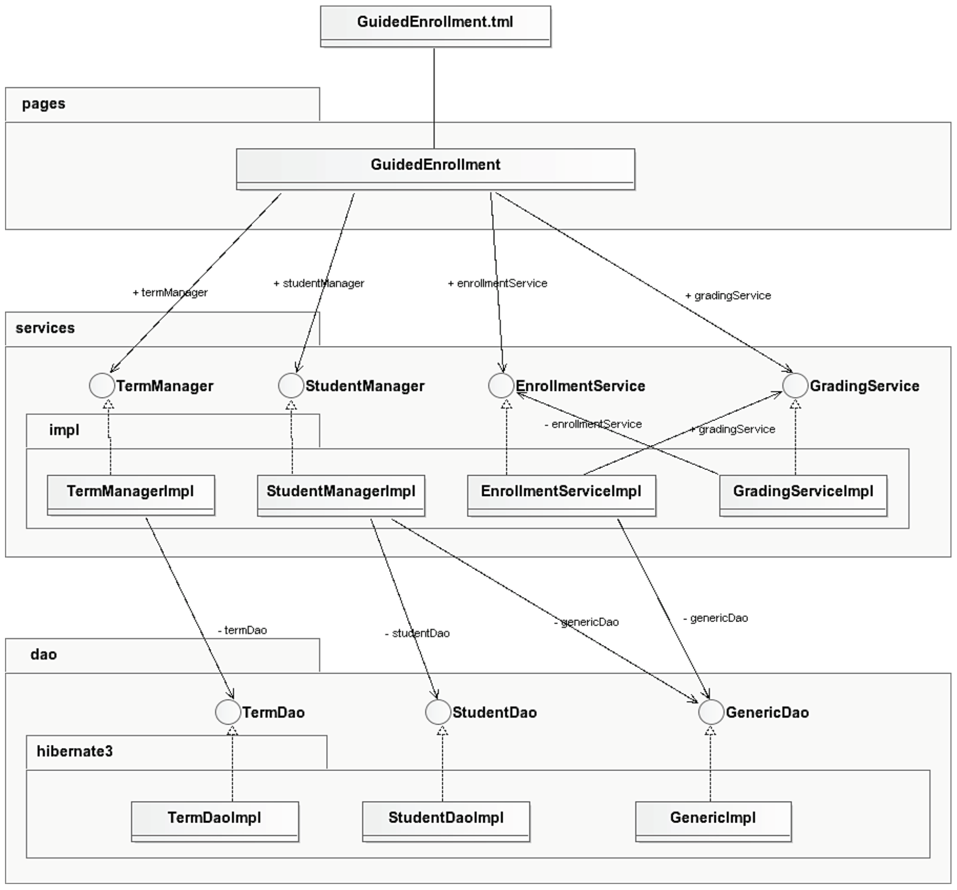


Figure 10. Organization of the internal architecture

Source: own elaboration.

- persistence classes – database access, storage and transactions, ORM, written in Java.

The connections and communications between most important classes of the sub-systems that sit behind the virtual academic adviser, are presented in Figure 10. Light coupling is used provided by the Tapestry IoC, so that any existing piece of the implementation can easily be changed with another.

4. Conclusion

The presented model defines a system that is able to change its structure, with traceability of all the modifications of the structure. At the same time, records are kept for the interaction of the visitors among themselves and with the system in association with the exact moment and context regarding how the structure of the system has changed.

This gives the possibility of introducing a self-adaptive feedback control loop, that the system will use to monitor itself, identify problems as symptoms and take actions in the form of slight modifications of the structure. The modifications are performed in control loop cycles which gives the ability to monitor the causality of the change of behaviour of the visitors and link this change back to a modification in the system, and so investigate if the structural change was an improvement or failure. In this way, the system can undo its steps and take counter measures against bad decisions.

References

Ajanovski V.V., *A personal mobile academic adviser*, [in:] *Mobile Web and Information Systems*, eds. F. Daniel, G.A. Papadopoulos, P. Thiran, *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 300-303. 2013a, http://link.springer.com/chapter/10.1007/978-3-642-40276-0_25 [accessed: August 31, 2013].

Ajanovski V.V., *Integration of a course enrolment and class timetable scheduling in a student information system*, "International Journal of Database Management Systems" 2013b, 5(1), pp. 85-95.

Ajanovski V.V., *Personalized adaptive system for term enrollments based on curriculum recommendations and student achievement*, [in:] *Proceedings of the International Conference Information Systems 2013*, IS-CONF, Lisbon, Portugal: IADIS International Association for the Development of the Information Society Press, 2013c, pp. 342-346.

Ajanovski V.V., *Towards a virtual academic adviser*, [in:] *Proceedings of the 8th International Conference for Informatics and Information Technology (CIIT 2011)*, Molika, Bitola, Macedonia, Institute of Informatics, 2011, pp. 146-149, http://www.getcited.org/pub/103508579 [accessed: 19.04. 2013].

Dieberger A., *Supporting Social Navigation on the World Wide Web*, 1997, https://smartech.gatech.edu/handle/1853/3525 [accessed: 22.01.2013].

Dobson S. et al., *A survey of autonomic communications*, "ACM Transactions on Autonomous and Adaptive Systems (TAAS)" 2006, 1(2), pp. 223-259.

Dourish P., Chalmers M., *Running out of space: Models of information navigation*, [in:] *Short Paper Presented At HCI*, 1994, pp. 23-26, http://fields.eca.ac.uk/deaua/wp-content/uploads/2008/10/hci94-navigation.pdf [accessed : 20.01. 2013].

Forsberg J., *Social navigation: An extended definition*, www. nada. kth. se/~forsberg/Documents/(letzter Aufruf am 12.08. 2007), 1998, http://www.nada.kth.se/~forsberg/Documents/SocNav.pdf [accessed: 22.01.2013].

Laddaga R., Robertson P., Shrobe H., *Results of the second international workshop on self-adaptive software*, "Self-Adaptive Software: Applications" 2003, pp.159-178.

Lemos R. de et al., *Software engineering for self-adaptive systems: A second research roadmap*, [in:] *Software Engineering for Self-Adaptive Systems*, eds. R. de Lemos et al., Dagstuhl Seminar Proceedings, Dagstuhl, Germany, Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2011, http://drops.dagstuhl.de/opus/volltexte/2011/3156.

Riedl M.O., *A computational model and classification framework for social navigation*, [in:] *Proceedings of the 6th International Conference on Intelligent User Interfaces, IUI '01*, New York 2001, pp. 137-144, http://doi.acm.org/10.1145/359784.360320 [accessed: 19.01.2013].

Spence R., *A framework for navigation*, "International Journal of Human-Computer Studies" 1999, 51(5), pp. 919-945.

MODEL I WDROŻENIE SAMOADAPTACYJNEGO MECHANIZMU NAWIGACJI W SIECIACH SPOŁECZNYCH W PUBLICZNYCH SYSTEMACH INFORMACYJNYCH

Streszczenie: W artykule przedstawiono model uogólnionego systemu nawigacji w publicznych systemach informacji, który umożliwia poprawę struktur i zawartości repozytoriów informacji poprzez samoorganizację opartą na mechanizmach nawigacji w sieciach społecznych. Podstawowym celem projektu jest opracowanie adaptacyjnego systemu nawigacyjnego. Model integruje pojęcia związane z nawigacją w sieciach społecznych, interakcją i adaptacją ze sprzężeniem zwrotnym. Model koncentruje się na elementach nawigacji w sieciach społecznych i wykorzystuje je w mechanizmie nawigacji, dostosowując go do indywidualnych użytkowników i otoczenia systemu. Wprowadzenie sprzężenia zwrotnego umożliwia rozszerzenie usprawnień w organizacji struktur informacyjnych. W artykule pokazano wykorzystanie tego modelu w uczelnianym systemie informacyjnym zarządzającym wyborem przez studentów przedmiotów z programu nauczania.

Słowa kluczowe: modelowanie procesu nawigacji, sieci społeczne, uczelniany system informacyjny, mechanizmy adaptacji.