INFORMATION
SYSTEMS IN
MANAGEMENT

Information Systems in Management (2014) Vol. 3 (1) 16–26

THE SYSTEM OF PLANNING AND MANAGEMENT OF THE MEDICAL SERVICES

DARIUSZ DOLIWA, WOJCIECH HORZELSKI, MARIUSZ JAROCKI

Faculty of Mathematics and Computer Science, University of Lodz

The system presented in this paper is designed to manage medical services at the Center for Rehabilitation and Cosmetology operating at the University of Computer Science and Skills in Lodz. The Center provides rehabilitation services and medical cosmetology. The purpose of the system is to create and modify the schedule of services provided by the Centre, including the availability of the resources, the preferences provided by the patient and the conditions resulting from the specific nature of these services. The system consists of the following components: a repository of resources, the databases describing the current schedule, the language of description of the relationships between the resources, a query language, with the editor supporting the query creation, and the two applications, the first generates a sequence of services required by the patient and the second modifies the current schedule to take account of the requested services.

Keywords: task scheduling, medical services managing, component oriented architecture

1. Introduction

There are many papers dealing with the topic of composition independent services based on semantic descriptions these services. Particularly noteworthy are the works [1-3]. The paper [4] provides an overview of the methodology of solving such problems. Some of these works consider static approaches, where flows are given as a part of the input, while the others deal with dynamically created flows.

One of the most active research areas is a group of methods referred to as AI Planning [5]. Several approaches use Planning Domain Definition Language [6]. The most complete specifications of an automatic composition system was described in [7]. The author proposed a solution based on a multi-phased composition using a uniform semantic description of services. Our approach uses similar methods. For service description was adopted language presented in papers [8-9]. The solution have been moved from the area of network services and implemented in a component-based business system.

2. System description

The system is designed to manage medical services at the Center for Rehabilitation and Cosmetology operating at the University of Computer Science and Skills in Lodz. The Center provides rehabilitation services in the form of medical visits, rehabilitation treatments and laboratory tests. This activity is not confined to individual patients, the majority of treatments are carried out to independent users of both individuals and institutions; the Center is also a service integrator for the external entities (branch offices, independent companies which want their services to be offered by the Centre). The purpose of the system is to create and modify the schedule of services provided by the Centre, including the availability of the resources, the preferences provided by the patient and the conditions resulting from the specific nature of these services. The planning involves searching for and allocation of the resources that meet the required needs. Designed for Center planning system should have the following characteristics:

- Users should be able to easily use the application provided by the Centre, have the ability to easily formulate their needs in a language close to natural (or by a simple graphical interface.)
- Resource allocation should be easily modified by employees without qualifications in the field of computer science.
- The system should be able to implement medical indications given as the result which is to be achieved (eg. a specific series of rehabilitation treatments of a particular type) rather than as a precise plan indicating where and by whom this goal should be pursued. Identifing the steps to achieve the goal should be automated by the system.
- Individual steps forming a plan can be implemented by a variety of the "service providers" (CIRK Center, cooperating entities etc.). The provided services may have the same functionality, but can vary according to the quality indicators (such as location, time, cost of the service). The System should allow to assess

the resulting plan and make adjustments by the operator (the resulting plan may be considered as a starting point for further manual adjustments that may be necessary during its implementation).

3. System architecture

The system consists of the following components (Figure 1.): a repository of resources, the database containing information about patients, doctors and physiotherapists and storing the current and past service reservations, the language of description of the relationships between the resources, the query language (with the editor supporting the query creation), and the two applications, the first generates a sequence of services required by the patient and the second modifies the current schedule to take account of the requested services.

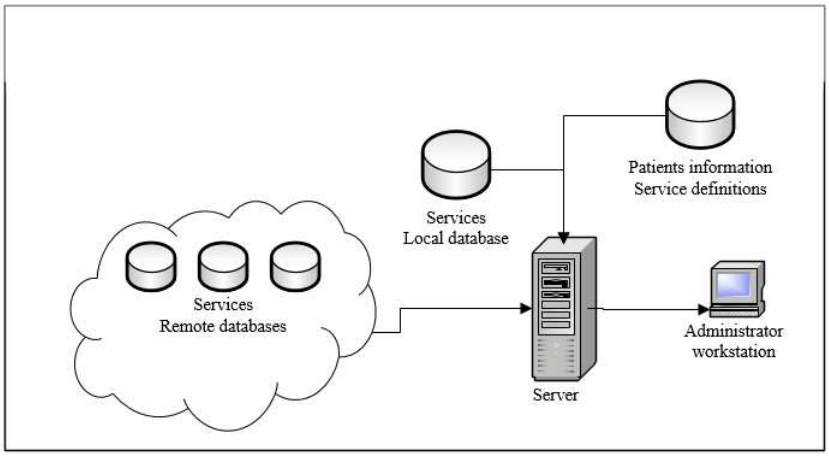

Figure 1. System architecture

A repository of the services for the system can be either a local database or the services obtained through the interface adapting the network services of the external providers. The repository allows you to query a list of the available treatments that meet the specified constraints, and returns the proposals of the service. The communication allows to reserve a service, in addition it is also possible to cancel a service (a patient gave up a treatment).

The language created for the application allows you to define the relationship between the objects available in the system (such as a patient, a treatment, a laboratory test, a physical therapist, a physician) and the services (a patient's registration, a medical examination, a laboratory test, a specified type treatment). It

also allows you to define the preconditions that must be fulfilled for the service to be performed, indicates how the objects participating in the service will be modified by the service and it also allows you to define the sequence in which the individual services will be performed.

The system contains also a query language that allows the patient to determine the required services and the conditions that they have to meet (eg. a location, the time of the service, a request for a specific doctor or a physiotherapist). Since the questions are to be constructed by the user, in the system there is available the query generator, which through a simple graphical interface allows the user to express his request in the system language.

After entering the user's query the application is run and generates a sequence of (or a set of the sequences) services that meet the needs of the patient and fill described by the language relationships between the offered services. Then there is verified their availability in the repository where the relevant services are reserved or we get the information that posed requests can not be processed. If the request can be made, based on the received plan, the current schedule of the tasks to be implemented by the Centre is modified.

4. System services

The application is designed to allow the end user (the patient) to schedule a visit to the Centre offering a wide range of medical services. It may be a visit to a general practitioner, a specific type of specialist doctor, execution of a single medical test or a group of tests, and finally planning one or more rehabilitation treatments.

List of available medical specialties, types of tests and types of treatments and their characteristics are defined by the service provider in the database. In the system we have objects, such as *Patient*, *Doctor*, *Physiotherapist*, *Test* and the services provided to the patient. These services are divided into types as a registering in the Centre (*Registration*), a medical examination (*VisitDoctor*), a laboratory test (*Therapy*) and a rehabilitation treatment (*Treatment*). Each of these services has some properties specified on the server (the service definitions). Each may have some preconditions for running and can cause changes in the system by creating new or changing the existing objects. The services may also leave during runtime some dynamic information about their use (*Trace* of service) allowing to save the information about the number of positions within a service sequence, time of the service execution, its cost, and other properties. In addition, this makes it possible to impose a particular sequence of services for the patient by the administrator.

An Object *Patient* represents the patient and has the following properties:

```
Patient := {id : int, name : last_name : string, first_name :
string,date_of_birth : date, pesel : string, phone : string,
sms : Boolean, diagnosis : string, symptoms : string }
```

It is created at the time of starting the use the system by the patient (via *Registration* service) and is a carrier of information about the patient and the history of its treatment.

Objects *Doctor* and *Physiotherapist* represent CIRK employees. Besides the basic information (personal information) they contain the data needed for planning the Center's work (such as an availability, a type of treatments carried out, etc.).

Objects *Test* and *Therapy* are similar in a structure and store information about tests performed in the Centre. They are created by the system administrator, if necessary - when the Centre plans to offer a new service - a test or a treatment. They have the following characteristics:

```
id     : int
name   : string
result : string
comment: string
```

The services offered in the system are administrative (*Registration*), or medical services (such *VisitDoctor*, *Test*, *Treatment*, etc..). A *Registration* service has no prerequisites and creates a new patient (an object *Patient*). It also sets the data of the patient's properties. It is a service that needs to be done once for each patient, before any other services. If the patient has already been created then this service will no longer be called by his subsequent visits to the clinic.
In the language of the system it can be described as follows:

```
Registration:
IN           :     null
OUT          :     p:Patient
PRECONDITION :     null
POSTCONDITION:     isSet(p.id) and isSet(p.name) and
                   isSet(p.last_name)and isSet(p.first_name)
                   and isSet(p.pesel)and isSet(p.phone)and
                   isSet(p.sms)
```

The other services offered by the system correspond to medical services. The objects of *VisitDoctor* type are services like *GeneralPhysician*, *Laryngologist*, *Cardiologist*, etc., representing a visit to a doctor of a particular specialty. Calling these services it requires the existence of a *Patient* object and sets the property diagnosis of this object.

The objects of *Laboratory* type are all kinds of laboratory tests (blood testing, DNA testing, etc.), and *Treatment* type are the treatments offered by the Centre (*Rehabilitation*, *Physiotherapy*, *Cytology* etc.).

The *VisitDoctor* and *Treatment* services have the following definitions:

```
IN                  :     p:Patient
OUT                 :     p:Patient and tr:TreatmentTrace
PRECONDITION        :     isSet(p.id)
POSTCONDITION       :     isSet(p.id) and
                          isSet(p.recommendation) and
                          isSet(tr.start) and isSet(tr.stop)
                          and isSet(tr.location)and
                          isSet(tr.servicename) and
                          isSet(tr.servicetype)
```

and they differed only by the values for the fields in the *servicetype* and *servicename*. The system assumes that the addition of a new medical specialty, new type of medical test or a new type of rehabilitation treatment requires the definition of the new services in the database on the server.

Planning a visit to the patient in the Centre will require the formulation of conditions of this visit in the system language (the query language), that determine the information held on the input (a requirement of the relevant objects and their properties) and the requirements to be met at the exit.

```
Input:
List of objects: . . .
Precondition: . . .

Output:
List of objects: . . .
Postcondition: . . .
```

An example scenario of a planned visit when the patient uses the system for the first time and would like to arrange a medical visit in order to determine the necessary treatments will look as follows:

```
Input:
List of objects:    null
Precondition:       null
Output:
List of objects:    p : Patient
Postcondition:      isSet(p.id) and isSet(p.name) and
                    isSet(p.last_name)and isSet(p.first_name)
                    and isSet(p.pesel) and isSet(p.phone) and
                    isSet(p.sms)and isSet(p.recommendation)
```

In this case, we do not require any object on the input, and we want to be created a new object representing the patient with properties that describe it and recommend further procedure. In more complex cases the query can be more complicated such as when a patient Kowalski (having already an account in the system and being assigned an identifier, has also staked a diagnosis) want to visit a dermatologist and perform blood test (the order of the sequence is important), in Lodz in the first week of January 2012. The query in this case looks as follows:

```
Input:
List of objects:    JKowalski:Patient,  T_GP:Trace
Precondition:       T_GP.service = ”General_physician”) and
                    isSet( JKowalski.diagnosis)
Output:
List of objects:    JKowalski:Patient,  T_GP: Trace; T_D1:
                    Trace, T_D2:Trace;
Postcondition:      (T_D1.servicename = ”Dermatologist”)  and
                    (T_D2.servicename=”BloodTest”) and
                    T_D1.level<T_D2.level and
                    month(T_D1.start)=”January” and
                    month(T_D2.start=”January” and
                    T_D.location=”Lodz”
```

In this example, we also apply the aforementioned object *Trace*, which allows you to control which services were developed earlier and for example determine the order of the services implementing the plan.

Writing queries in the system can be quite complicated for the person administering it (we assume that he is not a computer scientist). For this reason, in the system was created the query generator, which through a simple graphical user interface (Figure 2.) allows the user to express his request.

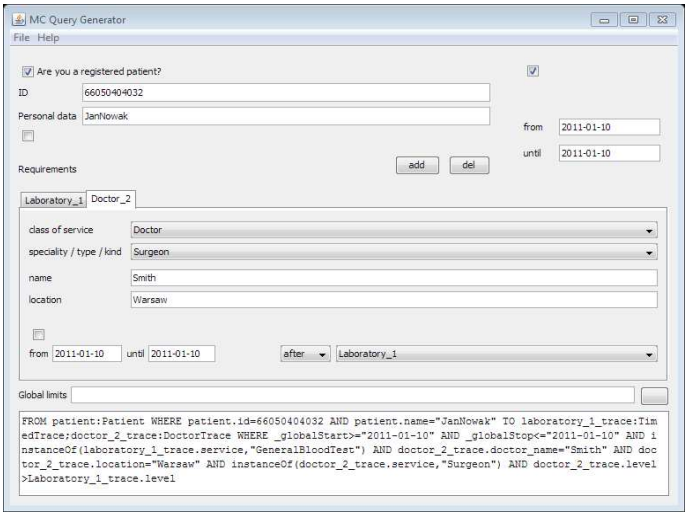

Figure 2. Query generator

The queries based on the service definitions will generate an initial plan which is a sequence of services implementing the patient's request. Then, for the resulting sequence must be associated with specific services (after checking their availability). For this purpose we need a repository of these services.

5. Repository of services

The services provided locally use the register in a database maintained on the server, but it is also possible for the patients of the Center to use the services from external sources which are provided by the web services. A record whether local or remote must allow for an inquiry (enables querying for available services without making commitments), a booking (filling in the patient for a specific treatment, carried out by a specific physiotherapist, in a given time and place) and cancellation (cancellation of a recording and release the associated resources).
A local database used in the Center has the structure shown in Figure Figure 3 and consists of tables such as Patients, Physioterapists, Doctors describing patients of the Center (the identity of the patient and contact details, symptoms and medical advice) and working in the Center doctors and physiotherapist (eg. personal data, represented specialties).

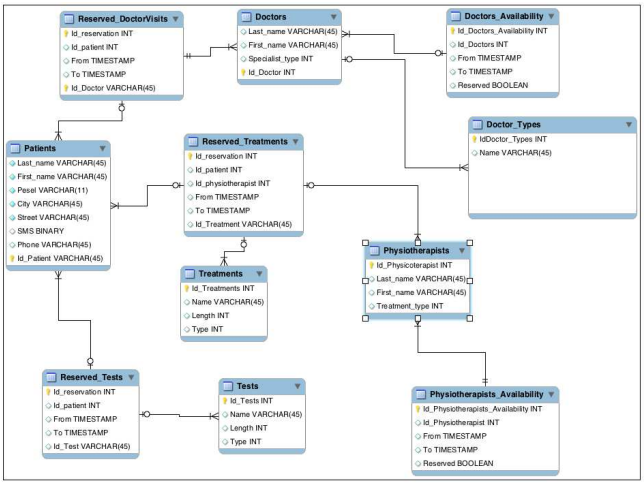

Figure 3. Database structure

The tables *Treatments, Tests* define the types of treatments and tests offered in the Center. Other tables are responsible for storing information about the availability of services and a staff, and are modified during creating a schedule of tasks. Additionally, the application, which generates a schedule based on these tables, selects the available services that meet patient's demand.

6. Plan generator

After defining a user's query the applications, which are a part of the system, create a plan that satisfies the conditions resulting from the query. The operations of generating the plan have two phases. In the first, we make a sequence of types of services that match the query, created on the defined dependencies between the services (eg preconditions services require the existence of objects created by other services) or defined directly by the user (through the references to objects *Trace*). For some more general questions we may have multiple correct sequences and then the user of the system is responsible for choosing one of them. In the second phase for the existing sequence the specific services are looking for in the repository (requests for a quotation mode), which is used by the system. They have already defined the properties such as a place of execution, a person responsible for their performance, the time in which they are available and the cost assigned to them . Including these characteristics and their compatibility with the query arises the actual plan of the services that can be offered to the patient. As in the first phase, if

there are several correct plans (corresponding to the query) the user is responsible for the selection. The selected services are marked as reserved in the repository and included in the schedule of tasks to be implemented by the Centre.

7. Final remarks

The system is a specialized implementation of the concept, which can be applied to various domains, enabling to build an integration system for distributed services of a common characteristic. More generally, a similar system can be implemented in every domain in which we have to plan an access to some resources. The system is based on concept of PlanICS [9], which was adapted to medical services modelling and scheduling.

A further contribution of the system is in an extended language of queries, enabling to express more requirements occurring is practice.

REFERENCES

[1] J. Rao. *Semantic Web Service Composition via Logic-Based ProgramSynthesis.* PhD thesis, Dept. of Comp. and Inf. Sci., Norwegian University of Science and Technology, 2004.

[2] J. Rao, P. Küngas, and M. Matskin. *Logic-based web services composition: From service description to process model.* In Proc. of the IEEE Int. Conf. on Web Services (ICWS'04). IEEE Computer Society, 2004.

[3] D. Redavid, L. Iannone, and T. Payne. *OWL-S atomic services composition with SWRL rules.* In Proc. of the 4th Italian Semantic Web Workshop: Semantic Web Applications and Perspectives (SWAP 2007), 2007.

[4] B. Srivastava and J. Koehler. *Web service composition - current solutions and open problems.* In Proc. of Int. Workshop on Planning for Web Services (at ICAPS 2003), 2003.

[5] M. Klusch, A. Gerber, and M. Schmidt. *Semantic web service composition planning with OWLS-XPlan*. In Proc. of the 1st Int. AAAI Fall Symposium on Agents and the Semantic Web, pages 55–62. AAAI Press, 2005.

[6] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins. *PDDL - the Planning Domain Definition Language - version 1.2.* Technical Report TR-98-003, Yale Center for Computational Vision and Control, 1998.

[7] S. Ambroszkiewicz. enTish: An Approach to service Description and Composition. ICS PAS, Ordona 21, 01-237 Warsaw, 2003.

[8] D. Doliwa, W. Horzelski, M. Jarocki, A. Niewiadomski, W. Penczek, A. Półrola, and M. Szreter: *Web Services Composition - From Ontology To Plan By Query*. Control & Cybernetics 40(2) pages 315-336, 2011.

[9] D. Doliwa, W. Horzelski, M. Jarocki, A. Niewiadomski, W. Penczek, A. Półrola, M. Szreter and A. Zbrzezny: PlanICS - *A Web Service Composition Toolset.* Fundamenta Informaticae 112(1), pages 47-71, IOS Press, 2011.

[10] D. Doliwa, W. Horzelski, M. Jarocki, A. Niewiadomski, W. Penczek, A. Półrola and J. Skaruz: *HarmonICS - a Tool for Composing Medical Services.* Postproc. of the 4th Central European Workshop on Services and their Composition (ZEUS'12), CEUR Workshop Proceedings vol.847, pages 25-33, 2012.