Architecture of an Active Life-Event Portal: A Knowledge-Based Approach

Anamarija Leben[1] and Marko Bohane[1,2]

[1]University of Ljubljana, Faculty of Administration, Gosarjeva 5, 1000 Ljubljana, Slovenia
anamarija.leben@fu.uni-lj.si
[2]Jozef Stefan Institute, Ljubljana, Slovenia
marko.bohanec@ijs.si

Abstract. We propose architecture of an active life-event portal, which draws on knowledge-based concepts and methods. The portal consists of three main modules: logical, data and control. They operate on three levels: topics, life-events and services. We present the functionality and operation of the modules, and propose knowledge representation methods for each level: (1) hierarchical trees for the level of topics, (2) modified eEPC for the level of life-events and (3) modified AND/OR graphs for the level of services.

1 Introduction

Life-event based web portals are developed and organized according to the realistic assumption that most people in a particular life situation do not know exactly which public services they need. For instance, a person only knows what he/she wants to achieve - to build a house, to start a business, to get married, etc. These situations are known as *life-events*. The web portal is supposed to have the necessary 'knowledge' to determine services and administrative procedures that are needed to solve the user's situation. Usually, various administrative procedures (resulting in public service delivery) at different administrative bodies have to be carried out. Thereby, a system that guides the user through the situation and helps him/her to identify the required services and their providers is needed. The web portal that includes such a system is called a *life-event portal* [5].

There are two types of life-event portals. The first, *passive life-event portals*, are based on a hierarchy of topics and life-events. The system allows the user to select topics and subtopics and in this way guides him/her to a particular life-event. When the life-event is selected, the information about public services and the necessary assistance is offered. Examples of such portals are Austrian Internet Service HELP (http://www.help.gv.at) and British UKonline (http://www.ukonline.gov.uk). There is faultiness in such approach; namely, the same life-event (e.g., starting a business) can differ in some aspects from user to user (e.g., starting an import/export business differs from opening a boutique with clothes). The problem is therefore that an individual life-event offers services regardless of the actual user's problem.

M.A. Wimmer (Ed.): KMGov 2004, LNAI 3035, pp. 147–156, 2004.

The second, user-friendlier life-event portals are called *active life-event portals*. The core component of such portals is a knowledge-based system: a computer-program based on inference mechanisms to solve a given problem employing the relevant knowledge [3]. The knowledge-based system in an active life-event portal uses a pre-defined structure of particular life-event to form an active dialog with the user. In this way, the user is involved as an active partner in the overall process of identifying and solving problems related to particular life situations. This approach therefore offers public services that better correspond to user's requirements than in the passive life-event approach. Singapore e-Citizen (http://www.ecitizen.gov.sg) portal uses such a system within 'starting a business' life-event.

In this paper, the architecture of an active life-event portal, designed within the project *Development of an intelligent life-event portal* in Slovenia, is presented. The project focuses on the development of an intelligent electronic guide through life-events employing the knowledge-based approach. The architecture of the system, its components and basic principles of its operation are presented in section 2. In section 3, methodological aspects of the proposed architecture are presented with special focus on the presentation and selection of formalisms for knowledge representation at various levels of the system. Section 4 concludes the paper.

2 Architecture of an Active Life-Event Portal

The primary goals of an active life-event portal are threefold [11]:

1. The identification of *life-event* applicable to the user's situation and requirements. This can be achieved using a hierarchical structure of topics, an annotated list of life-events, appropriate search mechanisms, etc.
2. The identification of *services* needed to solve a particular life-event. The list of services as well as the order of their application is determined *actively* on the basis of user's answers to particular questions in this decision-making process. The resulting list exactly corresponds to the user's life-event.
3. The identification of an *instance of each service* in the list. This also is a decision-making process that leads to the identification of specific input parameters needed to determine each service instance. For example, these parameters represent different *documents* that have to be provided by the user. For each service instance, the system provides suitable *guidelines*. The list of services with corresponding documents and guidelines represent invoking parameters for service delivery.

These identification tasks are based on data that is either provided by the user, either available in various *databases*. Databases can be *internal* (implemented as part of the portal) or *external* (e.g., public data bases or web services, accessible by the portal using network connections). Therefore, the portal must additionally support: (1) data acquisition from the user through dialogues and forms using a suitable user interface, and (2) data acquisition from internal and external data sources.

In order to fulfill these requirements, we propose architecture of an active life-event portal (Fig. 1). The architecture draws on knowledge-based concepts and methods. The portal consists of three main modules: logical, data and control. The following subsections describe these modules and their operation.

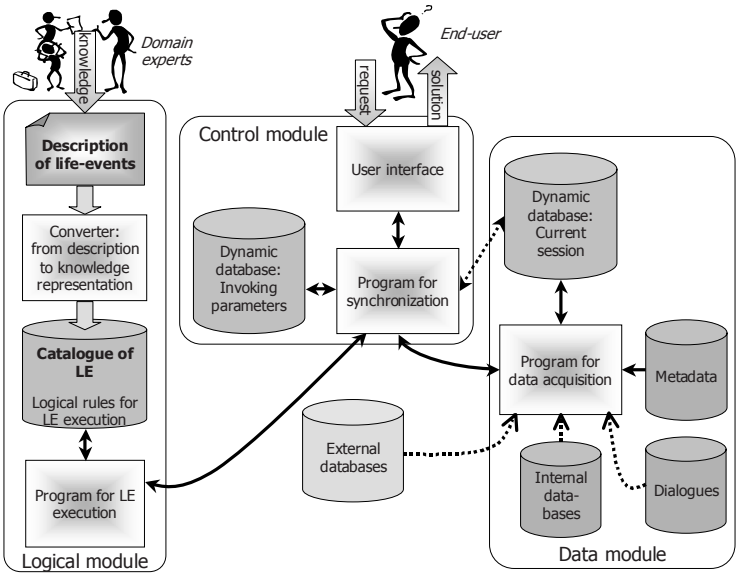

Fig. 1. Architecture of an active life-event portal

2.1 Modules

The basic task of the *logical module* is: given the user's situation and requirements, find the corresponding life-event and the corresponding lists of services, documents and guidelines. The life-event can be identified using a hierarchical structure of topics, which is presented to the user, who searches it and eventually finds the appropriate life-event. An alternative way would be using a suitable searching mechanism based on keywords. The lists of services, documents and guidelines can be then identified using: (1) logical rules, implemented within this module, and (2) user's data obtained from the data module.

In order to facilitate these tasks, the logical module should contain declarative and procedural knowledge about life-events. Declarative knowledge refers to static data about life-events and related objects, whereas procedural knowledge refers to rules of life-event processing. Specifically, the knowledge base of the portal should consist of the following components stored in the *catalogue of life-events*:

1. static description of the *hierarchy* and *properties* of topics, life-events and services,
2. rules for determining the list of *services*,
3. rules for determining the list of *documents* and *guidelines*.

The logical module takes care of the execution of these rules; therefore, in addition to the knowledge base, the module also includes *a program for the execution of life-events* – an inference engine.

It is important that the catalogue of life-events is designed independently on how the user's data is acquired, i.e., in which order, from which sources and through which

dialogues and entry forms. Namely, taking care about all these details may immensely complicate the rules contained in the knowledge base. Furthermore, this would require frequent changes of the knowledge base, for example, even after minor changes of the user interface. Thus, all tasks related to data acquisition are delegated to the data module.

The task of the *data module* is the acquisition and storage of data describing the current user and his/her life-event. Functionally, it serves the logical module. Whenever the inference engine of the logical module needs a particular data item, it makes a request to the data module. The data module first looks if this data item has already been acquired, and in this case transfers it immediately back to the logical module. Otherwise, the data module tries to obtain the item, either through the user interface invoking a suitable data entry form or dialogue, either by accessing an internal or external database. Thus, for each data item a method of its acquisition should be defined. These definitions are also part of the data module.

For facilitating these tasks, the data module should contain the following components:

1. *metadata*: static description of data items and their properties, such as: item name, unique identification code, data type, data source (method of data acquisition),
2. definition of *dialogues* and/or *entry forms* for the communication with the user; in general, each form contains several related data items and is invoked whenever the first data item in this form is requested by the logical module,
3. *internal database*, which contains data available locally, e.g. data about services, service providers and applicable laws,
4. definition of methods for accessing *external databases* and other external sources,
5. *dynamic database* containing already acquired data about the current user and/or the current session with the user.

The role of the third module, *control module*, is integrative. It executes the user interface, and controls and synchronizes the execution of the remaining two modules. It also collects the results of processing in an internal dynamic database, which include the identified life-event and the corresponding lists of services, documents and guidelines. The dynamic operation of this module, which invokes the other two modules, is described in subsection 2.2.

For both the logical and control module, we can determine three levels that correspond to the three basic goals of the portal and through which the portal guides the user from the identification of the problem to its solution:

1. *level of topics*, which serve for the identification of the life-event,
2. *level of life-events*, which facilitates the identification of services for solving the life-event,
3. *level of services*, which facilitate the identification of specific parameters needed to provide the services; these parameters include different documents that must be attached to the user's application; for each service, the system offers corresponding explanation and guidelines.

Using this viewpoint, the logical module describes these levels, whereas the control module takes care of the execution of tasks corresponding to each level.

An important additional part of the logical module is the support of *knowledge acquisition*. Buchanan [2] defines knowledge acquisition as 'the transfer and

transformation of potential problem solving expertise from some knowledge source to a program'. According to the proposed architecture of an active life-event portal, this process has two steps (Fig. 1):

1. With the help of domain experts, life-events are described in the way that is comprehensible to both domain experts and knowledge engineers and which can be easily transformed into some knowledge representation formalism. The module that supports this task (*description of life-events*) is part of the logical module.
2. A special program (converter) is then used to automatically transform this description into selected knowledge representation formalism implemented in the catalogue of life-events.

2.2 Operation of the System

The operation of the portal is initialised by the user's request for solving his/her problem. The first task of the system is to identify the corresponding life-event. The processing starts in the control module, which forwards the request for life-event identification to the logical module. There, the inference engine invokes rules for life-event identification. The execution of rules in turn invokes the user interface, showing the hierarchy of topics and life-events, allowing the user to select the right life-event. The control module saves the selected life-event into its dynamic database and in this way completes the first task.

The second task is to identify services for solving the user's life-event. For this purpose, the control module activates the service identification rules contained within the logical module. Actually, it activates the inference engine, which starts executing these rules. Whenever the rules require a specific data item, such as the user's date of birth or citizenship, a request is made to the data module in order to obtain it. The data module acquires the data according to its internal rules of data acquisition. If the item has been acquired previously, it is immediately sent back to the logical module. Otherwise, the item is acquired either from the user (invoking an appropriate entry form through the control module and its user interface) or by accessing an internal or external database. The data item is saved into the dynamic database of the data module for further use, and returned through the control module back to the logical module.

Based on obtained data, the rules identify services for solving the user's problem. Whenever a new service is identified, it is sent to the control module, which saves it into its dynamic database. In this way, the list of services for solving the life-event is gradually developed and saved within the control module. The list represents the sequence of services and if necessary, indicates services to be invoked in parallel as well.

The third task of the system is the identification of documents and guidelines. For each service contained in the list developed in the previous stage, the control module activates the inference engine of the logical module. Similarly to the previous stage, but using a different set of rules, the logical module processes this request and returns to the control module the list of required documents and instructions.

Finally, the control module presents the developed lists to the user using a suitable presentation. Typically, the system shows the list of services with adequate documents and guidelines.

3 Methodological Aspects of the Proposed Architecture

The gains and benefits that can be obtained from a life-event approach in public service provision are largely determined by the design of life-events and their presentation to the user. An analysis of existent life-event portals [6] shows that life-event is mostly understood only as a topic that simply incorporates different services, what corresponds to a passive life-event approach. In the proposed architecture, a life-event is understood as a *process* that includes *decision-making*, providing a list of services that best suits the real user's problem.

This viewpoint reflects in the methodology developed for the description of life-events and knowledge representation in the catalogue of life-events. The methodology has to deal with the modeling of *processes* (sequential and parallel processing of services), modeling of *data* (user data and data sources), and modeling of *knowledge* (knowledge about life-events). A result of decision-making is not just one service or decision whether a customer is eligible to a public service or not, but *a list of services* with some additional information. Therefore, different candidate formalisms and modeling techniques were considered for developing an adequate methodology:

- *process modeling*: data flow diagram, flowchart, extended event-driven process chain (eEPC), Petri nets;
- *data modeling*: entity-relationship (ER) model, object oriented modeling, semantic networks and frames, XML data model, first-order predicate logic;
- *knowledge representation*: declarative knowledge (property lists, semantic networks, frames, first-order predicate logic), procedural knowledge (production rules, decision trees, AND/OR graphs), uncertainty in knowledge (certainty factor in production rules, belief (Bayesian) networks, fuzzy sets).

In the following, some of these techniques and formalisms are discussed.

An *extended event-driven process chain* (eEPC) models three viewpoints of a process [9]: process, data and organizational viewpoint. It is suitable for modeling a life-event as a process. In addition to this, decision-making can be represented with an appropriate use of logical connectors. The problem with this representation is that it is not explicitly evident which function in a chain defines a decision and which one defines either a service to be included in a list of services, or some other life-event.

Petri nets [7,10] model a process as a sequence of transitions and places. The method allows to model parallelism and decisions (in a way similar to eEPC). High-level Petri nets allow the linking of process and data view. In our problem domain, they can be used for life-event description, but have the same shortcomings as eEPC.

Production rules [1,6] are presented in a simple 'if-then' form: IF condition THEN conclusion1 [ELSE conclusion2], where a condition of one rule can represent a conclusion of some other rule or a simple fact. A desirable feature of this formalism is that each rule defines a small and relatively independent piece of knowledge. Moreover, they can be easily modified by adding a certainty factor to represent uncertainty in knowledge.

A *decision tree* takes as input an object or situation described by a set of attributes and returns a decision – one of the predicted output values for the input [8]. A decision is reached by performing a sequence of tests from a tree root to one of the leaves. In life-event description, this would mean that a result is a single service; this is insufficient for life-event definition. Moreover, conditions in life-events are mostly

independent, what leads to a combinatorial explosion of decision tree. In our opinion, decision trees are therefore inadequate to model life-events.

Representation with *AND/OR graphs* relies on the decomposition of problems into sub-problems. This is advantageous if the problems are mutually independent and can be solved independently of each other [1]. Therefore, AND/OR graphs are suitable to present life-events and identification for service parameters as well. The main problem is that they do not depict a sequence and parallelism what is important for life-events description.

A *semantic network* represents knowledge as a network of relationships among entities, usually in a graph form, where nodes correspond to entities while relations are shown as labeled directed arch between nodes [1,3,6]. A *frame* is an object, designed to represent knowledge about an entity with slots (attribute-value pairs). The data structure within a frame resembles a small semantic network. Both formalisms use a mechanism of inheritance as an inference method and therefore can be used to model a hierarchy of topics and components of life-events.

3.1 Knowledge Representation in the Catalogue of Life-Events

As already mentioned, logical rules for determining a list of services and parameters of services represent procedural knowledge in the catalogue of life-events, whereas declarative knowledge includes static data about topics and their hierarchy, life-events and their components, services and their parameters.

Production rules were selected as formalism for procedural knowledge representation for the following reasons:

- adding new and changing old rules is relatively independent of existent rules;
- they can be relatively easily implemented in different technological environments (e.g. relational databases, Prolog language, any interpreted high-level programming or scripting language);
- they can be easily modified to represent uncertainty if needed;
- other representations, such as decision trees and AND/OR graphs, can be easily transformed into rules.

However, some modification of classic procedural rules was needed. To represent a sequence of services, a new type of production rules was introduced. Typically, the conjunction of conditions does not consider the sequence of testing and service execution; therefore, AND-THEN type of conjunction was added. A rule in the form 'IF condition1 AND-THEN condition2 THEN conclusion' means that first the condition1 and then condition2 has to be tested; that is, the order of services that fulfill conditions matters.

The main reasoning method to draw conclusions from knowledge in the catalogue of life-events is *backward chaining* (also called goal-driven reasoning) [6]. The main goal of the reasoning is: a life-event is solved if the main life-event rule holds. The conditions of that rule are usually conclusions of some other rules. In this way, a chain of rules is established for each condition in the main rule. These chains are searched using *depth-first search* resulting in a list of services and their parameters.

3.2 Description of Life-Events

In our opinion, a graphical form is especially suitable for the description of life-events as it is likely to be more comprehensible to both, domain experts and knowledge engineers, than some other techniques and formalisms. In the following, the selected techniques for each level of the system in the proposed architecture are described.

Level of topics

A model should depict a hierarchical structure of topics. A hierarchical graph (tree), an object model, semantic network and frames can be used for the modeling of topics. We chose a hierarchical graph as it most clearly depicts a hierarchical structure [11].

Level of life-events

For the description of life-events, the following features should be considered:

- *Decisions*: modeling of decisions,
- *Sequence*: modeling of sequence in execution,
- *Parallelism*: modeling of parallelism in execution,
- *List*: a result is a list of services,
- *Data, org.*: a model can be connected to a data and organizational viewpoints.

For each of these features, its *significance* for the description of life-events was assessed using the values: 1- unimportant, 2-important, 3-very important.

The following methods were considered: eEPC, Petri nets, production rules and AND/OR graphs. For each of these methods the *possibility* of modeling each feature was assessed using the values: 1-inadequate, 2- partly adequate, 3-adequate. In this assessment, two criteria were considered: whether a method enables the modeling of feature and how clearly a feature is represented in the method.

Table 1. Assessment of suitability of the methods for life-events description

		eEPC		Petri		IF-THEN		AND/OR	
Feature	Signif.	Possib.	Suit.	Possib.	Suit.	Possib.	Suit.	Possib.	Suit.
Decisions	3	2	**2**	2	**2**	3	**4**	3	**4**
Sequence	3	3	**4**	3	**4**	1	**1**	2	**2**
Parallelism	3	3	**4**	3	**4**	1	**1**	2	**2**
List	3	2	**2**	2	**2**	2	**2**	1	**1**
Data, org.	2	2	**4**	1	**3**	1	**1**	1	**1**
average suitability			**3,2**		**3**		**1,8**		**2**

Both assessments were aggregated into an assessment of *suitability*: how suitable is a method for modeling a feature, with following possible values: 1-unsuitable, 2-partly suitable, 3-suitable, 4-very suitable. The final assessment of the suitability of the method is calculated as an average of assessments for suitability of each feature. Table 1 presents final assessments: eEPC method and Petri nets were assessed as suitable, while production rules and AND/OR graphs were assessed as partly suitable.

We selected the eEPC method. To improve the suitability of modeling decision with eEPC, a new concept (decision point) was introduced; it combines a function, in which a decision is made, logical XOR connector and events describing alternatives of decision. *Decision points* are presented with diamonds, while their outputs (labeled directed links) represent *alternatives*. In our methodology, standard elements of eEPC diagram have the following meanings [11]:

- Functions (presented with rounded rectangles) are called *activity elements* representing *services* and different types of *life-events*. Each activity element is further described with its own model what leads to a hierarchy of models. Models on lower levels can be changed independently of the models on the higher level. In this way, we obtain modularity of the description of life-events.
- *Events* (presented with hexagons) help to control the processing of life-events: they either describe in which state the life-event is at a particular point of time, or identify the time at which a particular activity element is supposed to start.
- *Logical connectors* are presented with circles. With AND logical connectors, the parallelism in the processing of life-event is modeled, while OR connectors indicate alternatives.

Level of services

To define parameters for services, the following two features should be considered: (1) the modeling of decisions and (2) the modeling of combinations of parameters (e.g. supplements to application form). Therefore production rules and AND/OR graphs were considered as appropriate methods. We selected AND/OR graphs as they model combinations of parameters in a clear graphical form. In addition, a new concept similar to an OR node was introduced to represent a decision.

4 Conclusions

We have proposed a detailed architecture of an active life-event portal together with the description of its components, their operation and methods for knowledge representation. The main advantage of the architecture is in a relative autonomy of logical and data module enabling changing rules for life-event execution regardless of the way in which required data are obtained. This also holds for the data module: changing the way to obtain required data does not affect rules for life-event execution. The core module of the architecture is the logical module, which addresses all the key stages of the knowledge management lifecycle: (1) knowledge acquisition (the description of life-events, the converter), (2) knowledge representation (the catalogue of life-events), (3) knowledge use (the program for the execution of life-events together with the control module) and (4) knowledge maintenance (the program for the acquisition and description of life-events).

The logical module contains two distinctive components: (1) description of life-events and (2) catalogue of life-events. These two components effectively implement a two-step knowledge-acquisition process, which has the following advantages:

- all changes in a definition of life-events are made only in the description of life-events and are automatically transformed into the catalogue of life-events;
- the description of life-events is independent of technological implementation of the catalogue of life-events;
- the description is relatively independent of the knowledge representation formalism implemented in the catalogue as it supports different formalisms (e.g. production rules, decision trees and AND/OR graphs).

Considering three levels of the system, different methods of knowledge representation were selected for each level: (1) hierarchical tree for the level of topics,

(2) modified eEPC for the level of life-events and (3) modified AND/OR graphs for the level of services. The model of topics represents taxonomy for life-events classification. The life-event model considers three important aspects of life-events: (1) a life-event is a process defining the sequence, alternatives and parallelism in the execution of services; (2) it may include other life-events; (3) it involves a decision-making process to determine the list of services needed to solve the life-event. The model of services describes the combination of invoking parameters for the service.

A prototype portal based on this architecture is currently being implemented in Prolog. All the modules have been implemented together with the catalogue that includes two life-events: 'starting a business' and 'moving a home'. Further work will be focused on the testing and evaluation of this prototype, with special emphasis on testing the suitability of selected formalisms for knowledge representation. The final goal is to implement a web-based portal based on the proposed architecture.

References

1. Bratko, I. *PROLOG: Programming for Artificial Intelligence (third edition)*. Harlow: Addison-Wesley Publishers Ltd. & Pearson Educated Ltd. 2001.
2. Buchanan, B.G., Wilkins, D.C. (Eds.), *Readings in Knowledge Aquisition and Learninig.* Los Altos, CA: Morgan Kaufmann, 1993
3. Jackson P. *Introduction to Expert Systems (third edition).* Addison Wesley Longman Ltd., Harlow, (1999)
4. Leben, A., Bohanec, M. Evaluation of Life-Event Portals: Multi-Attribute Model and Case Study. In: Wimmer, M.A. (Ed.) *KMGOV 2003*, LNAI 2645, pp.25-36. Springer-Verlag. 2003.
5. von Lucke, J. Portale für die öffentliche Verwaltung: Governmental Portal, Departmental Portal in Life-Event Portal. In: Reinermann H., von Lucke J. (eds.): *Portale in der öffentliche Verwaltung.* Forschungsinstitute für öffentliche Verwaltung, Speyer, 2000
6. Mallach, E.G. *Understanding Decision Support Systems and Expert Systems.* Irwin Inc. 1994.
7. Reising W., Rozenberg G. *Lectures on Petri Nets I: Basic Models.* Springer-Verlag. 1998.
8. Russel S., Norvig P. *Artificial Intelligence: A Modern Approach (second edition).* Prentice Hall, Pearson Educational International. 2003.
9. Scheer, W. A. *ARIS – Business Process Modelling (third edition).* Springer-Verlag. 1999.
10. Sowa, J.F. *Knowledge Representation: Logical, Philosophical and Computational Foundations.* Brooks/Cole. 2000.
11. Vintar, M., Leben, A. The Concepts of an Active Life-event Public Portal. In: Traunmüller, R., Lenk, K. (Eds.), *EGOV 2002*, LNCS 2456, pp. 383-390. Springer-Verlag. 2002.