

# TOWARDS IMPLEMENTATION OF LIFE EVENTS USING GENERIC WORKFLOWS

**Mariusz Momotko**, Rodan Systems S.A., Warsaw, Poland

[Mariusz.Momotko@rodan.pl](mailto:Mariusz.Momotko@rodan.pl)

**Efthimios Tambouris**, CERTH/ITI and University of Macedonia, Thessaloniki, Greece

[tambouris@uom.gr](mailto:tambouris@uom.gr)

**Grzegorz Bliźniuk**, Military University of Technology, Warsaw, Poland,

Ministry of Interior and Administration of the Republic of Poland, Warsaw, Poland,

[Grzegorz.Blizniuk@wat.edu.pl](mailto:Grzegorz.Blizniuk@wat.edu.pl)

**Wojciech Izdebski**, Rodan Systems S.A., Warsaw, Poland

[Wojciech.Izdebski@rodan.pl](mailto:Wojciech.Izdebski@rodan.pl)

**Konstantinos Tarabanis**, CERTH/ITI and University of Macedonia, Thessaloniki, Greece

[kat@uom.gr](mailto:kat@uom.gr)

## Abstract

*The life-event concept is a meaningful metaphor aiming to reduce the gap between citizens' needs and provided public services. It enables public services to be composed in more sophisticated services that are able to satisfy more advanced citizens' expectations. Despite many recent efforts, however, implementing life events remains a challenge. Most of existing technologies that can be used to implement life events are either too static to effectively incorporate possible variations in citizens' needs and circumstances or too dynamic to be efficiently used by public administrations. In this paper we present an approach for implementing life events that is based on generic workflows technologies. This approach benefits from organising life events as processes (workflow management) and making them more flexible by including dynamic rules (rule management). To show suitability of the approach, we first determine the main requirements for implementing life events and then present how they are satisfied when adopting generic workflows technologies. The described functionality is illustrated with real life-event examples which are currently analysed in OneStopGov, a European Commission co-funded project.*

*Keywords: life event, one-stop government, generic workflows.*

## 1 INTRODUCTION

An increasing number of governments are considering a citizen-oriented approach to public service provision. In Europe, the i2010 eGovernment Action Plan (European Commission, 2006) suggests that “higher user satisfaction” and “convenient access to public services” should be priorities for all Member States. Similar statements are included in the National eGovernment Plans of many countries.

The life-event concept provides an intuitive metaphor for moving from a service-based organisation of public administration to a citizen-oriented approach. The use of this concept suggests organising public services around events that happen in the every day life of citizens e.g. getting married or travelling abroad. The fact that life-events provide an intuitive metaphor suggests they are particularly attractive for the presentation layer of one-stop government initiatives (M. Hagen and H. Kubicek,

2001; E. Tambouris, 2001) such as governmental portals (UK portal, 2006; USA portal, 2006; Austria portal 2006).

Indeed, life-events are currently used in a number of governmental portals in an attempt to provide a more intuitive interface, i.e. one interface where citizens would be able to access public services based on their needs, without explicit knowledge of the underlying fragmentation of the public sector.

Despite several attempts, implementation of life-events concepts to support one-stop government remains still a challenge. Unfortunately, most of the existing technologies that can be used to implement life-events are either too static to cope with possible variations of citizens' needs and circumstances or too dynamic to be useful for real cases.

For instance, using the standard workflow management and business management technologies it is hard to satisfy more dynamic requirements for processes in public administration (M. Weske and G. Vossen, 1998; S. Sadiq, 2000; M. Momotko and K. Subieta, 2002). Since all service variants and circumstances checks have to be expressed explicitly, definitions of workflow processes become very complicated. In addition, for some cases where the number of possible variants is not known at the definition stage (quite usual case for administration), standard workflow features fail. Finally, in more dynamic cases, in which there are some dependencies on execution history (previous cases or the current case) standard workflow management concepts are not sufficient too. A way to cope with such problems is to "program" workflow processes but in this case their manageability is weak and expensive.

On the other hand, technologies based on rule management are very flexible and seem to be able to satisfy all requirements for life-event implementation (V. Peristeras and K. Tarabanis, 2006). Using rule mechanisms, all required checks and situations can be expressed as pre-conditions and post-conditions of individual services. In this approach there is no direct definition of life-event. Instead, there are services which are invoked when their preconditions are satisfied. On the contrary, the number of possible service versions and options for real life events causes that the number of rules to be checked may be very large. In addition, at the definition stage it is quite easy to make a mistake and not including some conditions that avoid problems of execution order if more than one rule may be triggered at the same time.

In this paper we present an approach based on generic workflows and adaptability of workflow management systems. These terms were introduced by W.M.P. Van der Aalst (2001) and meant ability of workflow processes to respond effectively to dynamic changes either by making their definition more flexible or by modifying their execution dynamically. The proposed approach focuses on making the workflow process definition more flexible. It benefits from two technologies, namely workflow management and rule management. More specifically, life-events are defined as workflow processes while the dynamic elements of the life-event definition are expressed as rules. By following this approach, it can be argued that the flexibility of rules mechanisms is driven (controlled) by process definitions. That makes the approach applicable to real life events.

The paper is structured as follows. In section 2 we present the basic concepts including life event, public service and citizens' profile. On the basis of these concepts in the next section we report the most important requirements on life-event implementation. In section 4, we describe how these requirements may be satisfied by generic workflows. In the last section, we summarise our work in terms of advantages and open issues observed so far and present a plan for future work within the OneStopGov project (OneStopGov web page).

## **2 LIFE-EVENTS, PUBLIC SERVICES AND CITIZEN'S PROFILE**

In the technical literature, there is no universally accepted definition of life events. Amongst others, it has been suggested that "the term life event refers to the government services needed at specific stages in life" (European Commission, 2003) and that "life events describe situations of human beings where

public services may be required” (M. Wimmer and E. Tambouris, 2002). The main difference between the two approaches is that the first suggests that life events are in fact collections of public services (thus life events reside within the public administration domain) while the second suggests that life events are situations of humans (thus life events reside within the citizens’ domain).

In both definitions there is a close, yet unexplored, relationship between life events and public services. In addition, our analysis suggests there is a relationship between (a) life events perceived as citizens’ needs, (b) public services, and (c) the profile and circumstances of citizens. The profile and circumstances of citizens is particularly important in two cases: when mapping needs to services and when identifying the appropriate input document for a specific service. Each case is explained below.

Firstly, we have identified that the same need may require different public services based on the profile and circumstances of citizens. For example, the life-event “*I want to travel in Europe*” may or may not require invoking the public service “*issuing a visa*” depending on the “*nationality of the citizen*”, which is considered part of the citizens profile.

Secondly, we have identified that many public services contain a number of public **service versions (or variants)**, meaning that different input(s) may be required to execute a service again based on the profile and circumstances of citizens. As an example, the public service “*issuing a marriage permit*” is considered. This public service has a very large number of versions based on the profile of citizens. These include “*issuing a marriage permit [if one partner is a minor]*”, “*issuing a marriage permit [if one partner is an alien]*”, “*issuing a marriage permit [if one partner is a divorcee]*” etc. In all cases however the output is exactly the same i.e. the *marriage permit*.

For the purposes of this study a **life-event is a profile-based (personalised) set of actions, including at least one public service, which, when executed in its appropriate workflow, fulfils a need of a citizen arising from a new life situation.**

We now present a methodology for linking citizens’ needs with public services based on citizens’ profile. The methodology consists of two phases as shown in figure 1.

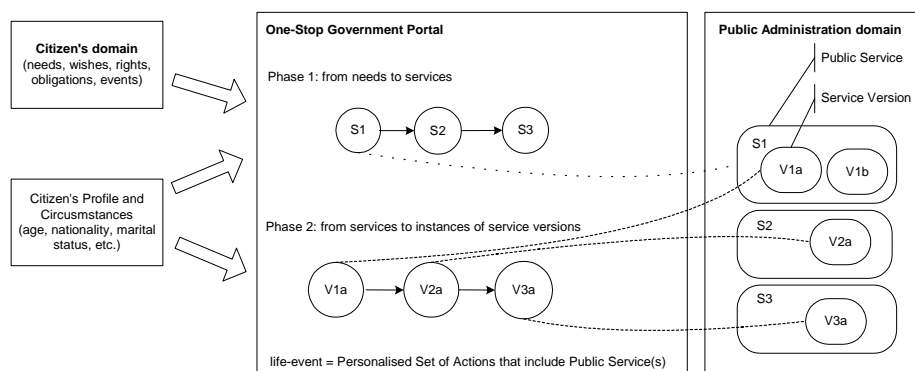


Figure 1. A methodology for mapping needs to services

### Phase 1. From needs to services

In this step the citizen’s need is analysed taken into account the citizen’s profile and circumstances. As a result, a set services is determined which when executed are expected to fulfil the citizen’s need. This set of services normally constitutes a workflow in the sense that there is an order (or dependencies) between the services. This workflow of services may be different for citizens that face the same need based on their profile and circumstances. As an example, consider the life-event “*getting married*”. In many countries, the relevant services are:

1. Public Service 1: invoke the public service “*issuing a marriage permit*”, then
2. Public Service 2: invoke the public service “*performing marriage ceremony at city hall*”, and then

3. Public Service 3: invoke the public service “*registering marriage*”.

## Phase 2. From services to instances of service versions

In this phase, each public service is examined vis-à-vis the profile and circumstances of citizen to determine the public service version that should be invoked and the precise service provider. In the previous example, the following public service versions are identified:

1. Version of Public Service 1: assuming that both partners are Greek, adults, live at the municipality of Kalamaria for the last two years, and were not married before then they should invoke the public service version “issuing a marriage licence” provided by “municipality of Kalamaria” with a certain number of inputs, then
2. Version of Public Service 2: partners should invoke the public service version “performing marriage ceremony at city hall” provided by “municipality of Kalamaria”, and then
3. Version of Public Service 3: one of the partners should invoke the public service “registering marriage” provided by “register of Thessaloniki” (since Kalamaria belongs to Thessaloniki).

## 3 REQUIREMENTS ON LIFE-EVENTS IMPLEMENTATION

In order to investigate the suitability of different technologies for implementing life events we will first determine relevant requirements. The requirements have been identified on the basis of the life-event concept introduced in the previous section. The identified requirements have been divided into functional and non-functional.

### 3.1 Functional Requirements

First, we focus on requirements for control flow within a life-event definition. A life event organises public services in a composition and may be considered as a **process**. In many cases, fulfilling a life event suggests invoking a number of public services **sequentially**. For example, the “*getting married*” life event consists of a sequence of three services: “*issuing a marriage permit*”, “*performing marriage ceremony at city hall*”, and “*registering marriage*”. In some other cases, fulfilling a life-event suggests invoking a number of public services **in parallel**. For instance, in “*setting a business*” life event, the services “*creation of the company stamps*” and “*register the company bank account*” may be executed at the same time.

Every public service includes all service versions and can be considered as a composed service or **sub-process**. Two service versions may be **alternative**. For instance, in Greece, if your partner is an alien, then to get married you have to use a service version which requires such documents as passport, certificate from country of origin of no legal independence to the marriage, and certificate that the partner is single. Otherwise, you have to use a service version which needs the other documents such as extract from birth register and identification document. Service versions may be also **optional**. For example, in Poland, for “*setting a business*” life-event, depending of the business’s location, the business owner have to pay taxes in one or in two different pay tax offices: VAT - in the tax office where your business is located, other taxes – in the place where the owner lives.

**Selection of a service version** depends on some information either taken from the **user profile** or provided as the **input** to start a given life event. In order to cope with such selection, definition of a life event should include a **generic specification of all service versions** which could be executed. This specification consists of the type of the public service in terms of its meaning (e.g. a service to register a marriage), its input and output parameters. When a life event is selected, the user profile and input data are used to determine concrete public services. This should be done before execution of the life event. For instance, if you set up a business, the tax office you will call depends on your address (given as an input to start a life event or taken automatically from your profile).

Some **user details** such as citizen's address may be stored **outside of the user profile**. In that case, the user profile stores just a reference to such data and uploads them when it is necessary in order to check circumstances for a public service. In addition, public services invoked in a life-event may require some data or documents provided by the other services, already executed within the life event. To cope with such input-output dependencies it is required to provide appropriate **data transformation mechanisms**. These mechanisms should be able to operate on simple types as well as **complex types**. In addition, for complex types they should be able to map a part of one data to other using standard query languages.

Once a life event is executed, both the citizen and the person who is responsible for management of the life event should be able to **monitor** it and have access to information on its **current state and execution history**. This information can be presented in a textual or graphical form and include such data as the names of the invoked services, start and finish dates, current status, etc.

Public services provided by public administration offices may be available via different protocols. Thus, the implementation of life events should offer a mechanism to **use various communication protocols** and add new invocation methods easily. These protocols should allow setting different levels of security and providing, for example, proper authentication of both communicating parties.

Finally, according to one-stop government, all **data/documents** required to execute a given life-event should be **identified before its execution**. To do that it should be possible to (a) **evaluate/simulate life-event definition**, (b) **ask all necessary questions** on data that are not reachable or may not be stored (e.g. because of civil rights) in the user's profile, (c) **determine what part of life-event definition is relevant to the profile and concrete needs** of the citizen.

### 3.2 Non-functional requirements

According to the law, there are **time constraints** on public services invoked within a life-event. Usually, a time constraint denotes the **maximum duration** for a given public service in order to produce a response. For example, getting a REGON number in Poland (i.e. statistical number for companies) from Statistical Office (i.e. invoke "get REGON" service) can last no more than 7 days. The other types of time constraints are related to **temporal dependencies between services**. For example, issuing a marriage has to be started no earlier than one month before marriage ceremony. Such dependencies are expressed by using timers and various control flow operators.

Another non-functional requirement is related to **security issues**. Sometimes it is not possible to store some information about citizens directly (e.g. address, contact information, identifiers) in a **user profile** if it is external to the provided public services. In such case, it should be possible to store in the user profile information (or a **reference**) what service should be used and what kind of other data the citizen needs to provide in order to get such information. In addition, the user profile may include some **data specific for processing of life-events** such as number of life-events executed, data of starting the first live-event, etc.

Since implementation of a life-event needs to include all possible cases, its definition may be very complex (in terms of invoked service versions, eligibility and circumstances checks). A life-event implementation should be **executed efficiently** in a real environment even though its definition is very complex.

## 4 GENERIC WORKFLOWS

Because of the 'process nature' of life-events, one of the most significant candidates to implement life events is **workflow management** technology. This technology treats life events as workflow (processes) and executes them according to their definition. Within a life event, a public service is represented as a composed activity (or sub-process) while a service version is represented as an atomic

activity. The order among services is represented by transitions. Circumstance and eligibility checks are represented by control flow elements such as split and join operations and transition conditions.

Many of the requirements for life-event implementations can be satisfied by **standard workflow management concepts**. These are: control flow requirements, data flow requirements (partially), dynamic specification of service types and monitoring requirements.

The other requirements such as advanced mechanisms to retrieve data, advanced time management, identification of required documents, questions for dialogue and part of life-event definition relevant to citizen's profile are related to more adaptive features and require extension of workflow management with **rule management** features.

Life-event implementation requirement	Required workflow management concepts	Applied workflow management and generic workflows concepts
<b>Functional requirements</b>		
Control flow, sequential execution	Standard	Control flow pattern, sequence
Control flow, parallel execution	Standard	Basic control flow patterns: parallel split and synchronization
Control flow, Alternative execution	Standard	Basic control flow patterns: exclusive choice and simple merge
Control flow, Optional execution	Standard	Advanced control flow patterns: multi-choice and multi-merge
Control flow, Sub-process	Standard	Supported in most process modelling and definition languages.
Data flow, data stored in user's profile	Standard	Supported by process data container.
Data flow, data referenced in the profile	Generic	Various methods to extract data from external resources based on business process query language.
Data flow, between services	Standard	Data mapping rules based on XML documents and XML transformations.
Monitor execution of life-events	Standard / Generic	Monitoring based on execution history. For visualization of execution history generic workflow features are required to represent history within the generic definition.
Generic specification of service versions	Standard / Generic	Application/adapters (services) executed within activities which are able to invoke the appropriate service version on the basis of received input parameters. If not enough, specification of concrete service version by flexible business rules written in a business process query language.
One-stop government approach	Generic	Validation of process definition, gathering required documents and necessary questions to be asked.
<b>Non-functional requirements</b>		
Time management	Generic	Advanced technique to manage time based on ePERT and constraints expressed in a process query language.
Assure security	Standard	Adapters executed within activities to assure appropriate level of security.
Efficient execution	Generic	Concretization / reduction of a generic definition before its execution

Table 2. *Specification of how life-event implementation requirements may be satisfied by workflow management technology.*

#### 4.1 Validation of Process Definition

Such extension is proposed by **generic workflows** which aim at increasing workflow process flexibility and adaptability in order to cope with complexity of process versions, as well as dynamic and evolutionary changes. To achieve these objectives, generic workflows suggest including dynamic rules in the process definition. These rules are dynamic since their validation is done not at the definition phase but at execution phase (and also during simulation). Usually such rules are used to define transition conditions, selection of invoked applications/services, and extraction of data from various (external) resources.

In table 2 we present how individual life-event requirements may be satisfied by workflow management technology. We also indicate whether the requirements can be satisfied by standard workflow concepts or require generic workflow concepts. The most advanced aspects of generic workflows are discussed more in detail in the consecutive sections.

One of the main ideas of one-stop government is to gather all required information from the citizen at one time before execution of the selected life event. This required information includes: (a) list of all questions which the citizen should be asked in order to determine what part of life-event definition is suitable for him/her, (b) list of all necessary data and documents which the citizen has to provide in order to start the life-event. The questions concern information that can not be extracted (directly or indirectly) from the user's profile.

To satisfy the above requirements generic workflows proposes to validate / simulate execution of the process which represents the life event. This validation will be illustrated on the example of *issuing a marriage permit* service. The definition of this service in terms of circumstances checks and all service versions is presented in figure 3. Depending on the circumstances (i.e. if the partner is alien and if he/she is below 18) there are four possible service versions to be called. All versions produced the same result, namely the *marriage permit*, but require different documents to be provided.

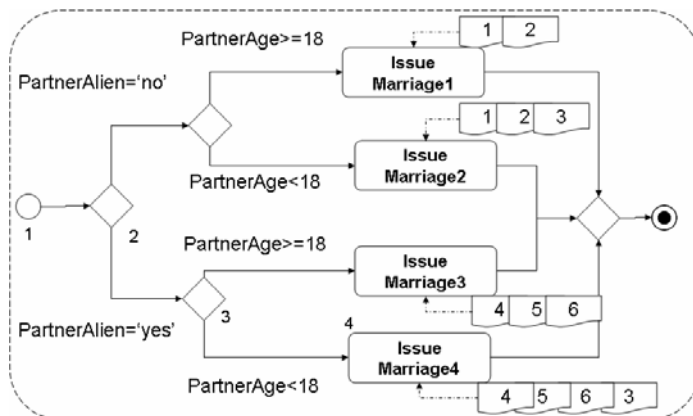


Figure 3. Definition of Issuing a marriage permit service<sup>1</sup>

The meaning of the required documents is as follows:

1. extract from birth register,
2. identification document
3. permission for getting married
4. passport
5. certificate from country of origin of no legal independence to the marriage
6. certificate that the partner is single

<sup>1</sup> We use Business Process Modelling Notation to model life-events and services.

Validation of this service starts from the starting point (1). Then the first decision is validated (2). Since this decision requires input data, the validation procedure collects the first question: “*Is your partner an alien*”? This question is displayed to the citizen. If the citizen answers “yes”, then the second decision (3) is validated. Please note that the upper versions are not validated because of the citizen answer. In the second diamond there is a need to ask the second question to the citizen: “*Is your partner below 18*”? If the citizen answers “yes” then the service version: “*Issue marriage permit 4*” is selected. Since this service requires four documents as an input, to start the live event the citizen needs to provide these four documents: D3, D4, D5, and D6. As the final result of such validation the systems returns to the citizen the list of documents that are required for his/her case to start execution of the life event.

As can be observed, extraction of the information about necessary documents is **driven by the process definition**.

#### 4.2 Generic-to-Concrete Process Transformation

Another problem is that generic workflows are quite complex and their execution may be hard and inefficient. To cope with this problem, before execution, generic workflows are concretised to a concrete workflow process. This transformation is done on the basis of process validation as it was described in the previous section. The parts of the individual services which will not be executed are removed from the process definition. In addition, all alternative decisions are also removed since just one of the possible options has been selected<sup>2</sup>. Figure 4 presents the reduced process for *issuing a marriage permit* example.

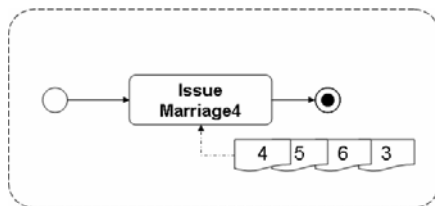


Figure 4. Reduction of issuing a marriage permit service based on life-event validation.

The reduced definition may be then represented in a standard process/service definition language, such as Business Process Execution Language for Web Services (WS-BPEL).

#### 4.3 Advanced Time Management

Generic workflows also put more focus on **quality of service management**, especially **time management**. One of the most promising methods to manage time is an **ePERT** method (see J. Eder, 1997; J. Eder et al. 1999). In ePERT, the authors propose a technique to model (temporal types such as duration and deadline), compute (timed graph construction) and verify time constraints (run time deadline calculation). The proposed workflow model is able to express sequential as well as parallel routing. For parallel routing alternative, conditional and unconditional routing elements are supported. In addition, optional activities are also supported. Construction of the timed graph consists of calculated E-values and L values for every process activity. These values are then used during process execution to indicate the timing status of a given process instance. Another approach for time management has been elaborated in the ADEPT project (P. Dadam, 2000). According to the ADEPT approach, for every activity four additional attributes corresponding to earliest and latest starting and

<sup>2</sup> In general, there can be more dynamic dependencies which prevent some part reductions.

finishing time are defined. To express more complex time dependencies between activities, so called time edges are used. The ADEPT system monitors process execution with respect to the calculated time constraints and informs performers if deadlines are going to be missed. Yet, another approach for time modelling has been presented by O. Marjanovic (2001). This approach is based on the idea of two level conceptual workflow modelling: the high level (or control flow level) and the operational level model. These two models are constructed to be semantically equivalent. The latter model is used for time modelling including modelling temporal constraints and verification of their temporal consistency.

#### 4.4 Flexible Business Rules

To express all possible variants and circumstances and keep the definition simple, generic workflows make process definition more flexible by including business rules that may be evaluated on the basis of external data (e.g. user's profile), history of previous executions (i.e. previous life-events) as well as the current execution (i.e. what services have been invoked so far).

The business rules may be used to express dynamic selection of concrete services from service versions (e.g. on the basis of the citizen's address). To do that business rules may use both internal (i.e. process data container, execution history) and external data (user's profile, references to another data available via services).

The business rules are usually expressed in a **business process query language (BPQL)**. BPQL queries are able to operate on all data provided either by workflow management systems or available from external resources such as user's profile. An interesting approach to BPQL has been proposed by M. Momotko and K. Subieta (2004). The authors proposed an object oriented language that is able to express all possible queries on a workflow process metamodel. Using BPQL in the context of life events, it is possible to ask service registry to retrieve an appropriate service implementation as well as to determine it on the basis of other functional and non-functional constraints. Since data, resources and applications can be stored outside of a workflow management system, BPQL is able to use data from user profile and other external sources. In addition, BPQL has a well defined semantics defined in terms of operations on a stack (i.e. environmental and query result stacks, see K. Subieta, 2004).

## 5 CONCLUSION

In this paper we analysed the concept of life events and we presented a first step towards implementation of life-events using generic workflows.

We started by defining a life event as a profile-based (personalised) set of actions, including at least one public service, which, when executed in its appropriate workflow, fulfils a need of a citizen arising from a new life situation. According to this definition, a life-event can be used to link needs to public services using the citizens' profile and circumstance.

We proceed by presenting a two-phase methodology that can be used for determining life-events. The first phase suggests that needs are mapped to public services. The second phase suggests that public services are mapped to instances of public service versions.

We thereafter proceed by listing a number of functional and non-functional requirements for life-event implementation. We also introduce generic workflows and present their applicability for life-event implementation illustrated also using the *getting married* life-event. The proposed approach supports one-stop government and is able to extract all necessary information on required documents and data from the citizen, before life-event execution. The approach aims at combining the strength of rules workflow management and rule management, and implementing life-events in a flexible and efficient manner. The main benefit of the approach observed so far, at this early stage of the OneStopGov project, is that generic workflows are able to include complete knowledge on all variants of a given life event. This knowledge may be then easily extracted and provided to citizens in order to explore

his/her concrete needs and then support him/her with complete information about required data and documents to realise those needs. In addition, from life event designing point of view, generic workflows assures that there is just one place where this knowledge needs to be expressed – life event definition. That may reduce drastically effort for maintaining knowledge on a life event.

We should note that according to the proposed approach, public administrations should think of their offerings in terms of service versions, and not just services. At the moment, existing public e-services can not be regarded as service versions. They usually hide the problem of versions by requiring multiple interactions with the citizen or by requiring just one composed document that includes all required documents for all versions. Since some documents are required just for one or several versions, this complex document needs to have many optional parts that are fulfilled only in particular cases (e.g. passport, if the partner is an alien). This approach suggests that the responsibility for verification of the provided documents (esp. completeness) is shifted to the citizen instead of doing it as automatically as possible on the system side. In addition, extracting information about required data before life-event execution is also very restricted if we use complex documents.

Future work will be focused on further validation of the conceptual framework on life events and the proposed methodology. We are currently analysing several citizens' needs, wishes, rights, obligations and events and relate them to public services. We are also analysing public services in terms of versions. Future work includes using this data to validate the conceptual framework but also to determine the citizens' profile and circumstances attributes i.e. those characteristics that the public administration should know about the citizen in order to provide him/her with personalised services. As the next step, we will examine whether and to what extent these attributes clash with civil rights, data protection and privacy directives etc. Finally, a prototype one-stop government portal platform will be developed and validated.

## Acknowledgments

The work presented in this paper was conducted within the project “A life-event oriented framework and platform for one-stop government” (OneStopGov) (OneStopGov 2006; E. Tambouris et al. 2006). The OneStopGov project is partially funded by the European Commission under the IST programme (contract number FP6-2004-IST-4 No 26965). The project has started on the 1st of January 2006 with an estimated duration of 30 months.

## References

- Aalst, W.M.P. Van der: *Generic Workflow Models: How to Handle Dynamic Change and Capture Management Information?* Computer Systems, Science and Engineering, 15(5), 2001.
- Austria portal, <http://www.help.gv.at>, accessed May 2006.
- Dadam P., Reichert M. and Kuhn K: *Clinical Workflows the Killer Application for Processoriented Information Systems*, 4th International Conference on Business Information Systems, BIS'2000, Poznan, Poland, 2000.
- Eder, J.; Panagos, E., Pozewaunig, H., Rabinovich, M., *Time Management in Workflow Systems*, 3rd International Conference on Business Information System (BIS'99), 1999.
- Eder, J.; Pozewaunig, H., Liebhart, W., *ePERT: Extending PERT for Workflow Management Systems*, 1st East European Conference on Advances in Databases and Information Systems (ADBIS'97), 1997.
- European Commission: *The Role of eGovernment for Europe's Future*, COM(2003) 567 final, Brussels, Sept, 2003.
- European Commission: *i2010 eGovernment Action Plan: Accelerating eGovernment in Europe for the Benefit of All*, COM(2006) 173 final, Apr, 2006.
- Hagen M., Kubicek H. (editors): *One-Stop-Government in Europe: Results of 11 national surveys*, University of Bremen, Bremen, [www.fgtk.informatik.uni-bremen.de/cost](http://www.fgtk.informatik.uni-bremen.de/cost), 2000.

- Marjanovic O: Methodological Considerations for Time Modelling in Workflows, 12th Australasian Conference on Information Systems, Australia, 2001.
- Momotko, M., Subieta, K., Dynamic change of Workflow Participant Assignment. 6th East European Conference on Advances in Database Information Systems, ADBIS'2002, Bratislava, Slovakia, 2002.
- Momotko, M., Subieta, K.: Business process Query Language – a Way to Make Workflow Processes More Flexible, 8th East European Conference on Advances in Database Information Systems, ADBIS'2004, Budapest, Hungary, 2004.
- OneStopGov home page, <http://www.onestopgov-project.org>.
- Peristeras V., Tarabanis K.: Reengineering the public administration modus operandi through the use of reference domain models and Semantic Web Service technologies, Proceedings of the 2006 AAAI Spring Symposium on The Semantic Web meets eGovernment (SWEG), Stanford University, California, USA, Mar, 2006.
- Sadiq, S.: Handling Dynamic Schema Changes in Workflow Processes, 11th Australian Database Conference, Canberra, Australia, 2000.
- Subieta, K.: Theory and Construction of Object-Oriented Query Languages, (in Polish) Polish-Japanese Institute of Information Technology, 2004.
- Tambouris E., Vintar M. and Tarabanis K.: A life-event oriented framework and platform for one-stop government: The OneStopGov project, Proceedings of Eastern European eGov Days Conference, 2006.
- Tambouris, E.: An Integrated Platform for Realising One-Stop Government: The eGOV project. In Proceedings of the DEXA International Workshops 2001, 359-363, Los Alamitos, CA: IEEE Computer Society Press, 2001.
- UK portal, <http://www.direct.gov.uk>, accessed May 2006.
- USA portal, <http://www.firstgov.gov>, accessed May 2006.
- Weske, M., Vossen, G.: Flexibility and Cooperation in Workflow Management Systems, Handbook on Architectures of Information Systems, pp 359–379. Berlin, Springer, 1998.
- Wimmer M. and Tambouris E.: Online One-Stop Government: A working framework and requirements, in Information Systems: The E-Business Challenge, (R. Traunmuller, Ed.), Kluwer Academic Publishers, pp. 117-130, 2002.