DOMAIN-SPECIFIC LANGUAGES FOR AGILE URBAN POLICY MODELLING

Michel Krämer
Fraunhofer Institute for Computer
Graphics Research IGD, Competence
Center for Spatial Information Management
Fraunhoferstr. 5, 64283 Darmstadt, Germany
Email: michel.kraemer@igd.fraunhofer.de

David Ludlow and Zaheer Khan
University of the West of England (UWE)
Faculty of Environment and Technology
Coldharbour Lane, Bristol BS16 IQY, UK
Email: david.ludlow@uwe.ac.uk
Email: zaheer2.khan@uwe.ac.uk

KEYWORDS

Urban policy modelling, Urban planning, Domain-Specific Languages, Human-computer interaction, Smart cities

ABSTRACT

In this paper we present a new approach of performing urban policy modelling and making with the help of ICT enabled tools. We present a complete policy cycle that includes creating policy plans, securing stakeholders and public engagement, implementation, monitoring, and evaluating a particular policy model. ICT enabled tools can be deployed at various stages in this cycle, but they require an intuitive interface which can be supported by domain-specific languages (DSLs) as the means to express policy modelling aspects such as computational processes and computer-readable policy rules in the words of the domain expert. In order to evaluate the use of such languages, we present a real-world scenario from the urbanAPI project. We describe how DSLs for this scenario would look like. Finally, we discuss strengths and limitations of our approach as well as lessons learnt.

INTRODUCTION

In general, a policy can be referred to as a plan of action adopted by an individual, department, organisation, business or government in a domain-specific problem context. A policy model is a descriptive or graphical representation of the plan of action. The process of developing a policy model can be referred as policy modelling. Ruiz Estrada reviews more than 1,500 scientific articles and discusses the evolution of policy modelling over the past three decades (Ruiz Estrada, 2010). He suggests its classification into 12 categories. According to him, policy modelling can be defined *"as an academic or empirical analytical research work that is supported by the uses of different theories, quantitative or qualitative models and techniques to evaluate the past (cause) and future (effect) of any policy implication(s) on the society anywhere and anytime."* Also his analysis reveals that there is a constant increase of econometric models in policy modelling and a significant lack of non-economic variables such as social, political, technological and natural factors simultaneously, which could increase vulnerability of policy modelling in the policy-making process.

In response to the sustainable development agenda and the rise of partnership-based urban planning, bottom-up policy development approaches, with increasing public participation for collaborative decision-making, are transforming the traditional top-down policy modelling approaches (Lempert, 2002). In particular, sustainable urban development necessitates ICT enabled tools to develop and demonstrate alternative urban models to different stakeholders and public for consultation as well as policy and decision-making. In this regard, the use of modern ICT enabled tools and techniques greatly improves the overall policy-making process. However, these ICT enabled tools require use of innovative technologies such as Web 2.0, 3D visualisation and simulations (Krämer and Kehlenbach, 2013) as well as mechanisms to support automated adoption of local action plans and proposed policy changes.

In addition to the above, techniques from the machine learning domain such as data mining or social mining can be used to refine policy plans based on feedback gathered from public participation or from the experience of actually implementing a particular policy (cf. Hanzl, 2007; Maragoudakis et al., 2011). Therefore, ICT enabled tools often need to perform rigorous analysis of data to take appropriate actions against different events. The tools are driven by operational or computational logic which needs to be defined by domain experts—i.e. urban planners and decision makers—who for that purpose typically have to use a general purpose programming language or scripting language such as Visual Basic.NET, Python, etc.

However, the real challenge can be related to expressiveness of these languages, intuitiveness and flexibility in defining machine-readable rules for domain-specific policies (Saleem et al., 2012). These languages are often hard to understand and to learn for non-IT personnel. In order to avoid requiring the domain experts to have a deep understanding of computer science or programming, a sophisticated, yet easy-to-understand user interface is needed. This interface should not be too generic but instead reflect application-specific and domain-specific issues. In this way, domain experts can focus on the actual problem instead of technical details.

Proceedings 27th European Conference on Modelling and Simulation ©ECMS Webjørn Rekdalsbakken, Robin T. Bye, Houxiang Zhang (Editors)
ISBN: 978-0-9564944-6-7 / ISBN: 978-0-9564944-7-4 (CD)

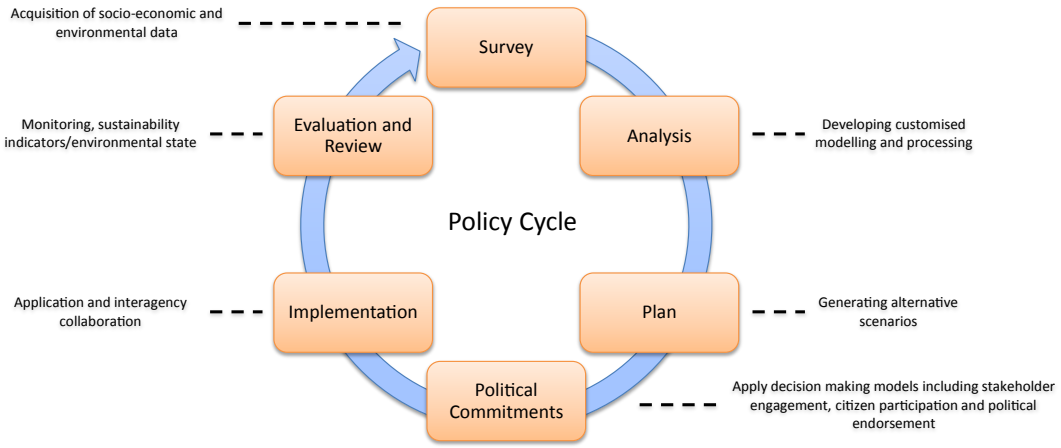

Figure 1: Policy development process

During policy modelling, urban planners often define rules that later provide the basis for policy plans or final policies (Guzy et al., 2008). For example, the municipality's aim to reduce car traffic in the city could be expressed with rules that set upper limits for the number of cars per hour on specific streets (see the real-life example scenario below). ICT enabled tools can automatically check if these rules are met or—semi-automatically—assist domain experts in defining alternatives or new policy rules. Typical ICT tools for that purpose are expert systems or more generally rule-based systems. Expressing machine-readable rules with such systems often requires the user to have a background in computer science. Apart from that, rule-based systems are typically rather generic and flexible and do not focus on domain-specific issues.

Hypothesis

In consequence of the above, in this paper we investigate a new way to express urban policy modelling aspects such as computational processes and machine-readable policy rules with so-called Domain-Specific Languages (DSLs). A DSL is a language that is tailored to a specific application domain. It consists of terms (vocabulary) from that application domain, so it is easily understandable by domain experts. At the same time, the DSL is also machine-readable and—depending on the actual use case—possibly executable. Consequently, a DSL can be used to allow users with non-IT background to communicate with the machine—i.e. to write computer programs or scripts, to declaratively model data, etc.

To summarise, our hypothesis for this research work is as follows:

"Domain-specific languages help urban planners to control ICT tools in the policy cycle and to focus on the actual problem—i.e. policy modelling—without requiring them to have a deep understanding of technical details."

Methodology

In order to evaluate the use of domain-specific languages for urban policy modelling, in this paper we first present the policy cycle which includes deploying policy plans, public participation, implementation and monitoring as well as gathering feedback and including it in future decisions. After that we present the current state of the art in domain-specific language design. In the second part of this paper we describe a real-world scenario from the urbanAPI project which is funded by the European Commission (FP7 RTD). We describe the different stages of the policy cycle that can be automated by ICT tools and where DSLs can assist domain experts in their tasks. We then show two example DSLs that are readable by both policy actors as well as machines. Finally, we discuss strengths, limitations and lessons learnt.

POLICY CYCLE

In order to support policy modelling using ICT, it is necessary to understand the policy-making process. Figure 1 depicts a generic policy process as a cycle representing different stages of the policy-making process (boxes). The process begins with the 'Survey' stage that collects domain specific data—e.g. socio-economic and environmental relevant to the issue of urban development, etc.—either by using surveys, polls, or ICT technologies—e.g. sensor nets, etc.—for problem or issue identification.

The next stage of the policy making process (Analysis stage) utilises data gathered at the survey stage, and provides an assessment of the territorial impacts, in respect of socio-economic and environmental variables, that identify the problem to be resolved by the plan.

The Plan stage is the formulation of a coherent strategy, specified by the technical administration experts (urban planners) in respect of a variety of policy objectives, that address the problems identified in the analysis stage, and which proposes a plan of action over a period of time (five to ten-year period) to resolve these problems.

In the Political commitment stage urban planners provide a proposition for future development of the urban territory typically subject to public and wider stakeholder consultation, following which a political commitment is

made by elected officials of the municipality to the implementation of the plan.

Implementation of the plan (in the Implementation stage) over the plan period of several years involves commitments by a variety of public agencies acting in concert to secure the objectives of the plan in order to respond to the problems identified at the survey/analysis stages and to provide a framework for private investment in the development of the urban area.

The Evaluation and Review stage is focused around the monitoring of the implementation of the plan to identify the extent to which the plan is achieving the objectives identified with the policies of the plan, and where it is failing to fully meet the policy objectives of the plan, to provide a basis for reformulation of the plan in the next stage of the policy cycle.

The process repeats in a cycle in order to assess and improve current policy implementation.

AUTOMATING POLICY MODELLING

From theoretical computer science point of view, policies are defined by rules—i.e. conditions and respective actions—that implement software business and computational logic—e.g. software security policies, etc. These rules can also be used for the policy development process (see the example rules from the real-life scenario presented below). We differentiate between three levels:

i) rules for defining the operational or computational logic for ICT tools used at different stages of the process;

ii) rules for describing policy restrictions and expectations that can be automatically checked by ICT tools or semi-automatically evaluated by domain experts with the help of ICT tools;

iii) rules to integrate process stages in order to automate the flow of information from one stage to the next stage.

The first level necessitates machine-readable domain-specific vocabulary with common semantics and reusable syntax that can be used by ICT enabled tools to collect and rigorously process the data, generate alternative scenarios, perform decision-making and monitor and assess the overall impact of the policy implementation. This information can be reused in the next policy development cycle.

The second level requires a vocabulary that is easy to comprehend for the domain expert and for decision makers or other stakeholders who—in respect to a bottom up policy modelling approach and a participatory process—need to understand the rules for the urban plans. At the same time the language must be machine-readable so it can be used for automatic rule evaluation. It therefore needs a well-defined grammar and syntax.

The third level necessitates defining interoperable interfaces between ICT enabled tools. This enables tools at different stages of the process to interact with each other and to facilitate flow of information from one stage to the next stage.

However, it is difficult to fully automate the policy development process due to necessary engagement of different stakeholders including citizens and policy-makers at different process stages. Nevertheless, ICT enabled tools can provide a semi-automated approach to support socio-technical interactions, data collection, modelling, processing and analysis, and visualisation of alternative domain-specific scenarios for collaborative decision-making (Batty, 2007). In response to the requirements of the three levels described above, we propose to use domain-specific languages as the interface to the user or domain expert. We will present an example use case below where we use DSLs for computational logic and for describing policy restrictions.

DOMAIN-SPECIFIC LANGUAGES

In computer science domain-specific languages are used for a number of purposes. For example, in the UNIX operating system configuration files are typically written in custom languages. In agile software development domain-specific languages are used to quickly adapt to changing user requirements. DSLs are also used for machine-to-machine communication. In the IETF protocol specifications, for example, textual, ASCII-based DSLs avoid platform-dependent details such as encoding issues that would normally arise with binary communication protocols.

In recent times, DSLs have gained a lot of interest, especially in the scientific community. One of the most actively pursued topics is DSL design. Mernik et al. differentiate between five phases of DSL development: decision, analysis, design, implementation and deployment (Mernik et al., 2005). The analysis phase is one of the most important ones since it includes specifying user requirements (Tairas et al., 2009). Mernik et al. identify three common ways to develop a new DSL:

- the DSL will be implemented based on an existing language which will be included completely (so-called internal DSLs are an example for this);
- an existing language will be limited to the means needed for the application domain;
- application-specific vocabulary and language constructs will be added to an existing language.

Apart from that, DSLs can of course be developed from scratch as well. Such languages are typically called external DSLs since they are not embedded into or based on an existing language. In his book "Domain-Specific Languages" Martin Fowler describes a number of methods for developing parsers for such languages (Fowler, 2010). The drawback of external DSLs is that you cannot rely on existing language constructs or compiler components. However, they allow for a much more flexible design

since they are not bound to the syntax and grammar of a host language and the possible restrictions implied by that. Based on this, the DSLs that we describe in the example use case below are external ones.

Since DSLs are used more and more often in modern software systems, there's a growing need for language maintainability. DSLs ought to help reduce maintenance costs in large software systems, but this can only work if maintaining the DSLs itself is not too costly. Therefore some work has already been done in the area of language modularisation. Hudak tries to reuse compiler components such as lexer and parser as well as semantic analysis (Hudak, 1998). Irazabal and Pons present a modularisation technique based on Xtext (Irazabal and Pons, 2010). They import several partial language definitions and merge them into a new single one.

We also identified the problem of maintaining DSLs as well as learnability and commonality as utterly important for our approach. More details on this can be found below in the section on 'building DSLs for urban planning'.

Xtext is a language workbench written in Java and based on the Eclipse Modeling Framework (EMF) and ANTLR as the underlying language recognition tool (http://www.eclipse.org/Xtext). It allows domain-specific languages up to general purpose languages to be defined. In recent times, Xtext has gained prominence and is used for a wide range of applications in the Java community—e.g. the Xtend language that adds useful features to Java, Spray which is a DSL for the Graphiti framework, etc. While Xtext is quite popular its main purpose is to quickly create DSLs for single, separated use cases. As we show later our approach depends on modularity and reusable language constructs. In our experience Xtext currently does not support this enough.

DOMAIN-SPECIFIC LANGUAGES FOR POLICY MODELLING

In this section we will present how domain-specific languages can be applied to the policy-making and implementation process (policy cycle), otherwise characterised as policy modelling. We will first describe an example scenario from the urbanAPI FP7 EU project (http://urbanapi.eu). After that we will discuss at what stages ICT enabled tools and DSLs can be used reasonably. Finally, we will present an example DSL that is both, readable by domain experts and by machines.

Example scenario

In the urbanAPI project, the city of Bologna, Italy is pursuing urban planning and environmental objectives. Khan and Ludlow describe the use case as follows (cf. Khan and Ludlow, 2011, pp. 43–55). The initiatives "Ambiente Vitale" and "Di nuovo in centro" aim for creating new public and green spaces as well as improving the mobility system. In this context, one specific area is of most interest. The San Vitale district is one of the oldest parts of the city with a rather heterogeneous and complex infrastructure. The area has a large population density of 17,464 citizens per km^2 (3,370 citizens in an area of 192,962 m^2). It is very close to the University of Bologna which makes it a popular area for student residence, the urban elite, and—given its location in the heart of the historic city—numerous tourists. In their spare time, citizens of the San Vitale district participate in committees and cultural associations which organise many events during the year. There are many commercial activities centred around the district's core mostly led by immigrants. To summarise, the district is heterogeneous in many ways, in its widespread, ancient infrastructure as well as in its population and cultural life. This has previously led to some oddities and infrastructure problems the municipality would like to avoid.

The urban planners are aware that the initiatives for restructuring this district require to involve all stakeholders. In order to enable public participation, the municipality plans to deploy ICT enabled tools including a 3D virtual reality (VR) application available on the Internet. This application will allow urban planners to present their plans to a wide audience and hence raise the awareness among citizens about future developments.

One central part of improving the mobility system in the San Vitale district is to reduce car traffic and instead extend public transportation by adding new bus routes for example. The city already owns a wide range of data that can be used for the 3D VR application including building footprints and heights, land use, coverage and zoning information. In order to estimate the consequences of changing the mobility system, the municipality plans to exploit GSM data (Global System for Mobile Communications). This data is gathered by mobile devices such as mobile phones, smartphones, tablet PCs or even car sensors. Within the urbanAPI project GSM data will be used to simulate traffic on the streets of Bologna. The simulation results will be included in the 3D VR application.

DSLs in the policy cycle

As described above, rules can be used at various levels: to define operational or computational logic, to describe policy requirements, and to provide platform-independent machine-to-machine communication between the different tools at the various stages in the policy cycle. In this section we will furthermore show at what stages DSLs can be used reasonably. Note that the different stages are rather complex in that they include not only data processing or visualisation but also statistical modelling, monitoring and human interaction. In the following we will only focus on tasks that can be supported by ICT tools and DSLs in particular.

In the scenario depicted above, the municipality of Bologna has already gone through the first stage of the cycle ("Survey"). They have gathered data such as building footprints and heights which can be used to create a simple 3D building block model. They also possess basic socio-economic, transport, utility, vegetation, census data, etc. In stage two ("Analysis") they therefore pre-process

the data. Here they can make use of automated, rule-based processes. A typical DSL that would drive such an automated process could look like this:

```
When there is a building footprint f
and there is a corresponding
    building height h
then extrude f by h.
```

This script would create a simple 3D building block model of the city of Bologna. It could drive a standard geographical information system (GIS) that manages the municipality's cadastre consisting of building footprints (typically 2D polygons) as well as respective building heights attached as metadata or attributes. GISs usually already provide a set of well-known geometrical functions like the polygon extrusion used in this script. Basically, the DSL terms map to GIS functions and parameters (or more precisely function arguments). The term '*there is*' maps to an object query based on its type ('*building footprint*'), '*building height*' maps to an attribute, whereas the term '*corresponding*' links the attribute to the respective building footprint. Finally, the term '*extrude*' maps to the polygon extrusion function whereas '*h*'—i.e. the building height—is the function's first argument.

The DSL used in this script is clearly readable and easily understandable. At the same time it uses a well-defined grammar and can thus be processed by machines. We use the keywords **When** and **then** here—instead of the more commonly used ones **If** and **then**—in order to mimic declarative production rules. Production rule systems are typically event-based, so that *when* something happens *then* the rule-based system will perform some specified action. In our case this means that we can keep the rule system running, so that *when* a new building footprint and a corresponding height is added to the data set *then* the 3D building model will automatically be created.

After preparing the data, the urban planners might want to define rules for their scenario. For example, they want to reduce traffic on street A. At the same time they do not want the traffic on street B—which is an alternative to A—to increase. Therefore they would like to close A for cars and create new bus routes in order to provide an alternative for using the car. In order to evaluate later if the policy has been implemented successfully, the municipality evaluates passively collected GSM (Global System for Mobile) data from active mobile phones for street A and B to measure the number of cars passing by during the day. Such functionality is provided by the Public Motion Explorer application developed in the urbanAPI project. Using a DSL, this task can be performed automatically by ICT tools based on the acceptance criteria defined by the urban planners. A DSL for these criteria could look like as follows:

```
The number of cars on street A per
   day was 3000.
The number of cars on street A per
   day has to be 0.
The number of cars on street B per
   day was 2000.
The number of cars on street B per
   day has to be lower than 2500.
```

The idea of defining acceptance criteria in such manner can also be found in computer science in the area of Behaviour-Driven Development (BDD) where DSLs help developers to create machine-readable, executable code that is also understandable by their clients—i.e. the domain experts (see North, 2009).

These acceptance criteria can now be used in the consecutive stages of the cycle: during stakeholder engagement and for monitoring. The 3D visualisation, for example, can be driven by these rules. The main purpose of this application is to simulate the policy's consequences by visualising the 3D building model in combination with information about traffic density based on GSM data. Whenever one of the rules above do not apply, the user will be notified. For example, we can use the following rule to update the display:

```
When the number of cars on street B
   per day is higher than 2500
then display street B in red.
```

Again, we are using a production rule here. Due to its event-based nature the display will automatically be updated whenever something in the GSM data set has changed or whenever a stakeholder has interacted with the system—e.g. by changing some of the parameters such as number of maximum cars per street.

During the public participation stage, involving engagement with urban stakeholders including citizens, machine learning techniques such as data mining and social mining can be used to analyse how stakeholders react to the policy plan (Hanzl, 2007). For instance, the municipality might use information from the Internet such as page views, comments, likes/dislikes, etc. (Charalabidis et al., 2012, p. 157) to analyse the public opinion and to gain feedback on the policy plan (cf. Maragoudakis et al., 2011). The results from this analysis can be used to improve the acceptance criteria. "Improving" means in this case adding new rules. For example, during stakeholder engagement one might notice that street C is also often used as an alternative to A. Hence, it might be useful to also consider C in the acceptance criteria:

```
The number of cars on street C per
   day was 500.
The number of cars on street C per
   day has to be lower than 600.
```

Furthermore, after the policy has been implemented and its consequences are monitored these rules can be used to automatically assess the policy's success. The results gained from this, which will be identified in the monitoring and plan evaluation stage of the policy process, can then be incorporated into another round of the policy cycle.

Building DSLs for urban planning

As described above, the DSLs presented in this paper are external ones. They are not based on an existing language and they are not embedded into a host language. The main advantage of this approach is that the DSLs can be designed independently and therefore better tailored to the use case and the user requirements. However, this also might necessitate several small DSLs for the different stages of the policy cycle. The example use case above contains at least two of them.

In order to avoid having domain experts to learn a whole lot of different languages, we suggest to use language modularization—as described, for example, by Hudak (1998) or Irazabal and Pons (2010)—and to define a common vocabulary. This makes sure domain experts recognize words, terms and expressions they already know from other languages which makes it a lot easier for them to learn a new one. At the same time, the languages we propose here can be used in different stages of the policy cycle. For example, the rules defined in the plan stage can later be used in the evaluation stage as well. We'd like to point out that the semantics underlying the rules stays the same between the different stages—the meaning of the rules does not change—but the execution might differ. In one case, the rules are used to change the visualization in the political commitment stage, and later exactly the same rules are used to automatically evaluate the policy.

A language's vocabulary is influenced by the domain it is used in as well as user requirements. In order to find a common, reusable vocabulary we propose to make use of techniques known from different areas:

- methods from the semantic web, especially ontology engineering to find domain-specific terms (concepts) and relations;
- object-oriented design, in particular methods to build a domain model;
- techniques from software requirements analysis such as interviewing, contextual inquiry or apprenticing.

An example how a DSL vocabulary can be engineered using these techniques is described by Mauw et al. (2004) and later by Tairas et al. (2009) who suggest to start with creating an ontology (or using an existing one if already available) and then applying additional formal domain analysis methods.

DISCUSSION AND CRITICAL ANALYSIS

The Bologna example, identified above, demonstrates the ways in which the complexities of the policy modelling process can be facilitated by ICT tools. 3D visualisation itself provides a major advance in communication between the policy-making community (urban planners and the political domain) and urban stakeholders, including citizens (Al-Kodmany, 2002). The representation of plan proposals for a particular locality, in this case urban neighbourhood, in a real-life (3D) manner offers a significant advance on previous two-dimensional and plan based communications in which misunderstandings by the urban stakeholders regarding the nature of the policy proposals have been widely reported.

However, ICT enabled policy modelling is a complex process that deals with multiple variables. It consists of socio-technology interactions and often it is difficult for the end-user community to understand the complexity of underlying IT tools and languages. In order to facilitate end-user community engagement, DSLs are identified to define rules for policy development using an intuitive language and interface.

The Bologna example shows that DSLs can indeed be utilised to drive the ICT enabled policy cycle. Scripts and rules written in such a language are easy to understand for domain experts, because they use a domain-specific vocabulary. At the same time, they are machine-readable, since the DSLs have a well-defined grammar and syntax.

DSLs also have some drawbacks (cf. Axelsson et al., 2009). Since they are tailored to a specific application domain and sometimes only to a single use case, they are also rather limited regarding their expressiveness and reusability. A DSL designed for traffic simulation provides means for this specific application only. This means that the same DSL cannot be used for any other use case.

The aim of using DSLs is twofold. On the one hand, they consist of a limited vocabulary which makes them easy to comprehend and to learn. On the other hand, the mere number of different DSLs can overtax users if for each application they have to learn a new language. The key to designing reusable DSLs is to find a good balance between uniqueness of the language—i.e. how much it is targeted to a specific use case—and commonality. In particular, using a common vocabulary for several similar languages will make them easier to learn. Good language design of multiple DSLs therefore starts with defining a basic set of words and expressions from which the more specific languages can then be derived.

From an urban planning perspective, it is frequently argued that the urban planning is highly specific to each locality defined by national and local legislative provisions, structural frameworks and procedural requirements that greatly limit the development of common ICT solutions that enhance the plan making and plan implementation process. This local specificity of urban planning requirements for ICT development in support of urban planning is certainly evident. Nonetheless, the opportunity for the development of generic ICT tools supporting urban planning is evident in the fundamental proposition that cities and towns of Europe, and indeed globally are subject to common drivers of change (economic, social and environmental) that invite and benefit from common solutions.

Common solutions based on generic ICT applied at the local level offer benefits in numerous ways, not least from the perspective of multi-scale governance, in enhancing communication between the levels of government,

whereby upper levels can communicate more effectively with lower-level organisations that have common information requirements and provide common information outputs. Generic and common ICT solutions applied to the cities of Europe also create substantial market opportunity promising procedural efficiencies and cost minimisation.

The generic form of the policy cycle and its stages suggests capturing multiple models/view points—e.g. complex socio-technical interactions between different actors and entities, information model, activity model, etc. Also, each stage itself presents a specific view point suitable for certain type of stakeholders. For instance, the 'evaluation and review' stage should generate necessary information for policy makers to make a decision. Such a multi-model structure heavily relies on a common domain specific vocabulary that not only helps in defining common concepts that can be used across models and stages of policy cycle but also underpins a DSL for communicating across multiple models and stages of the policy cycle. However, in this paper we focused on using DSLs for describing computational logic for ICT tools used at different stages as well as for modelling policy restrictions that can be automatically or semi-automatically checked by domain experts with the help of ICT tools. Communicating across multiple models and stages remains a topic for future research.

CONCLUSION

In this paper we presented a new approach to model urban policies using domain-specific languages (DSLs). These languages can be used to drive ICT tools which assist urban planners during policy making. We presented a complete policy cycle consisting of different stages including plan making, public participation and evaluation. Through a real-life scenario from the urbanAPI project we demonstrated how DSLs can be used reasonably in the various stages of the policy cycle. Scripts written in such DSLs help domain experts to control the ICT tools without requiring them to have a deep understanding in computer science or programming.

Sustainable urban development is transforming the relationship between the urban planning community and urban stakeholders, requiring enhanced engagement and partnership based urban planning. The ICT enabled urban planning applications discussed in this paper provide a major opportunity to deliver the required participatory bottom-up urban planning. Full realisation of this opportunity is critically dependent upon the transition from the current heterogeneity of urban planning systems to one where generic ICT modules can successfully operate. The key to this is twofold. First, the policy-making cycle and model of plan production and implementation is in itself generic. The policy-making cycle utilises different information sources, according to different locally defined procedures, but fundamentally the policy-making cycle is common to all planning agencies.

The second requirement in order to create the opportunity for generic ICT urban planning applications is the recognition that there are many pathways to the desired result of good urban planning. In other words, substitution of certain current working practices by alternative procedures at the local level will permit the adoption of generic ICT solutions and thereby the necessary transformation of urban planning as a stage in the attainment of fully collaborative sustainable development.

In addition, we think the success of generic ICT tools greatly depends on intuitive user interfaces. DSLs as they are presented in this paper can provide such interfaces. They can drive the automated policy cycle and at the same time they are understandable by stakeholders who can therefore clearly see the decisions behind urban policies.

Basically, DSLs have been used quite successfully in computer science already to build intuitive user interfaces that are understandable by non-IT experts. However, until now they have not been applied to the area of urban policy modelling and making. We consider this paper the first step towards supporting the participatory, bottom-up approach to urban planning, and intuitive interfaces (i.e. DSLs) the key factor to its success. However, concrete application of this approach including a practical meta-model and a common vocabulary for urban planning DSLs remains a topic for future research.

ACKNOWLEDGEMENT

Research presented here is partly carried out within the project "urbanAPI" (Interactive Analysis, Simulation and Visualisation Tools for Urban Agile Policy Implementation), funded from the 7th Framework Program of the European Commission, call identifier: FP7-ICT-2011-7, under the grant agreement no: 288577, started in October 2011.

REFERENCES

Al-Kodmany, K. (2002). Visualization tools and methods in community planning: from freehand sketches to virtual reality. *Journal of planning Literature*, 17(2):189–211.

Axelsson, E., Sheeran, M., Stenström, P., Dévai, G., Horváth, Z., and Vajda, A. (2009). Domain Specific Languages: state of the art and future directions. *Ericsson Software Research Day, Stockholm, Sweden*.

Batty, M. (2007). *Cities and complexity: understanding cities with cellular automata, agent-based models, and fractals*. The MIT press.

Charalabidis, Y., Triantafillou, A., Karkaletsis, V., and Loukis, E. (2012). Public policy formulation through non moderated crowdsourcing in social media. In Tambouris, E., Macintosh, A., and Sæbø, Ø., editors, *Electronic Participation*, volume 7444 of *Lecture Notes in Computer Science*, pages 156–169. Springer Berlin Heidelberg.

Fowler, M. (2010). *Domain Specific Languages*. Addison-Wesley Longman.

Guzy, M. R., Smith, C. L., Bolte, J. P., Hulse, D. W., and Gregory, S. V. (2008). Policy research using agent-based modeling to assess future impacts of urban expansion into farmlands and forests. *Ecology and Society*, 13(1):37.

Hanzl, M. (2007). Information technology as a tool for public participation in urban planning: a review of experiments and potentials. *Design Studies*, 28(3):289–307.

Hudak, P. (1998). Modular domain specific languages and tools. *Fifth International Conference on Software Reuse*, pages 134–142.

Irazabal, J. and Pons, C. (2010). Supporting Modularization in Textual DSL Development. *XXIX International Conference of the Chilean Computer Science Society*, pages 124–130.

Khan, Z. and Ludlow, D. (2011). urbanAPI Deliverable D2.1: User Requirements Definition, version 032.

Krämer, M. and Kehlenbach, A. (2013). Interactive, GPU-based urban growth simulation for agile urban policy modelling. In *Proceedings of the 27th European Conference for Modelling and Simulation (ECMS)*.

Lempert, R. (2002). Agent-based modeling as organizational and public policy simulators. *Proceedings of the National Academy of Sciences of the United States of America*, 99(Suppl 3):7195–7196.

Maragoudakis, M., Loukis, E., and Charalabidis, Y. (2011). A review of opinion mining methods for analyzing citizens contributions in public policy debate. In Tambouris, E., Macintosh, A., and Bruijn, H., editors, *Electronic Participation*, volume 6847 of *Lecture Notes in Computer Science*, pages 298–313. Springer Berlin Heidelberg.

Mauw, S., Wiersma, W. T., and Willemse, T. A. C. (2004). Language-driven system design. *International Journal of Software Engineering and Knowledge Engineering*, 14:625–664.

Mernik, M., Heering, J., and Sloane, A. M. (2005). When and how to develop domain-specific languages. *ACM Computing Surveys*, 37(4):316–344.

North, D. (2009). Behaviour-Driven Development. Retrieved February 9, 2013, from `http://behaviour-driven.org/`.

Ruiz Estrada, M. A. (2010). What is Policy Modeling?. *SciTopics*. Retrieved February 8, 2013, from `http://www.scitopics.com/What_is_Policy_Modeling.html`.

Saleem, M., Jaafar, J., and Hassan, M. (2012). A Domain-Specific Language for Modelling Security Objectives in a Business Process Models of SOA Applications. *AISS: Advances in Information Sciences and Service Sciences*, 4(1):353–362.

Tairas, R., Mernik, M., and Gray, J. (2009). Using ontologies in the domain analysis of domain-specific languages. In Chaudron, M. R. V., editor, *Models in Software Engineering*, volume 5421 of *Lecture Notes in Computer Science*, pages 332–342. Springer Berlin Heidelberg.

AUTHOR BIOGRAPHIES

MICHEL KRÄMER is deputy department head of the Spatial Information Management competence center of the Fraunhofer Institute for Computer Graphics Research IGD in Darmstadt, Germany. His research interests are in Compiler Construction, Language Recognition and Artificial Intelligence as well as Big Data and Cloud Computing. He's development lead of the 3D GIS area and has contributed to various open source products. As the Scientific Manager of the urbanAPI project he's responsible for software architecture and the project's scientific progress. Michel Krämer holds a master's degree in computer science from the THM University of Applied Sciences, Gießen, Germany where he's now also a lecturer. His email address is `michel.kraemer@igd.fraunhofer.de`.

DAVID LUDLOW is member of the EU Expert Group on the Urban Environment, and the EU Expert Group contributing to the preparation and definition of the EU Thematic Strategy on Urban Environment (6th Environment Action Programme). He gained detailed knowledge of the subject area of European urban and regional environmental planning and sustainable development as well as the development of information, communication and technology (ICT) applications for sustainable urban management. He is responsible for the development and implementation of more than 40 major EU funded research projects, as well as publications of pan European significance. His email address is `david.ludlow@uwe.ac.uk`.

ZAHEER KHAN is postdoctoral Research Fellow in the Faculty of Environment and Technology of UWE and holds Bachelors, Masters and PhD degrees in computer science. He has over 10 years of experience in academic research and teaching. His research interests are use of ICT technologies for smart cities, urban management and policy modelling. His main expertise lies in the application of the state-of-the-art technologies from distributed computing, grids, clouds, sensor web, software engineering, business process management, data management, software agents, and geospatial information systems in multi-disciplinary application domains. He has been working on the large-scale collaborative SAGE multi-agent system, FP6 Health-e-Child, FP6 HUMBOLDT, FP7 LifeWatch projects, and is leading requirements specifications and evaluation work package in the FP7 UrbanAPI project. His email address is `zaheer2.khan@uwe.ac.uk`.