# Modelling Process Intensive Scenarios for the Smart City

Riccardo Cognini, Flavio Corradini, Andrea Polini, Barbara Re

Computer Science Division, School of Science and Technologies
University of Camerino, 62032 – Camerino (MC), ITALY
{riccardo.cognini, flavio.corradini, andrea.polini, barbara.re}@unicam.it

**Abstract.** Smart city can be considered as a process-intensive environment that needs to be as flexible as possible to support a continuously evolving scenario. In this paper we present an approach to support flexibility of Business Processes regulating the behaviour of ICT systems deployed within a smart city. The approach permits to deal with large collections of process variants thanks to the integration of Business Process notations and Feature Model descriptions. The approach is applied to a smart mobility scenario with a specific focus on bike sharing systems.

## 1 Introduction

The smart city foresees the efficient integration and usage of resources and services in order to improve quality of life [1]. Smart city is a complex eco-system where enabling infrastructure supports application scenario such as mobility, energy, health, etc. In such a context each application scenario can be implemented starting from resources and services that are strictly related to city characterization (i.e. number of inhabitants and weather forecast). It also means that the same system can be implemented in different ways according to specific needs (i.e. the bus transportation system can be integrated or not with rail transport solutions). In such a context citizens and companies expect that Public Administrations provide added value services to be used in many different scenarios. The supporting IT infrastructure and related organizational aspects have to be conceived to be flexible in order to be aligned, with laws, the overall smart city policies, and objectives. These aspects are submitted to continuous changes in order to make the smart city idea more and more effective.

In response to such dynamic scenarios each smart city can implement different variants of the same Business Process (BP) to support a large set of interactive services. As a result BP variability becomes an important characteristic of any smart city initiative, in particular when IT systems are considered, and they need to be kept aligned with emerging requirements. In such line of research this paper intends to illustrate how a complex system, typical of a smart city scenario, can be modelled and represented taking into account possible variability points. In particular variability is represented thanks to a novel modelling approach that integrates BP notations and techniques used within the software product

line community. The proposed approach has been validated according to a real scenario such as the Bike Sharing System (BSS).

The paper is organized as follows. Section 2 presents some background material, and then Section 3 gives an overview of smart city services characteristics exemplified on the BSS. Section 4 describes the proposed approach and then Section 5 introduces more details on the application to BSS. Finally, Section 6 presents relevant related works, while Section 7 draws some conclusion and opportunities for future work.

## 2 Background

### 2.1 Business Process Management and Business Process modeling

Business Process Management (BPM) "includes concepts, methods, and techniques to support the design, administration, configuration, enactment, and analysis of Business Processes" [2]. "A BP is a collection of related and structured activities undertaken by one or more organizations in order to pursue some particular goal" [3] with or without an electronic/digital support. In the smart city, BPs have to consider operations across services and organisations. Process model provides a horizontal view of the business focusing on connections, handovers and the responsibility for what happens between organisations. Starting from such integration and confirming the role of technology, it is important to take the point of view from the perspective of those who will use the service or of those will participate to its provisioning. Smart city public-private services structure, their input and output, the interdependencies among different elements can be supported by notations and tools supporting the BP abstraction.

The accuracy of the BP modeling phase is critical for the success of an organization in particular in scenarios in which it is necessary to adapt to changing requirements. In order to design a BP different classes of languages have been investigated and defined.

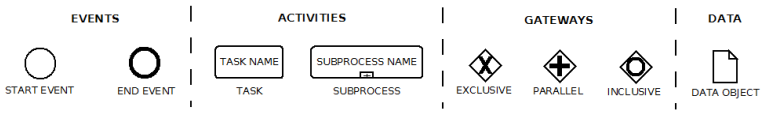


**Fig. 1.** BPMN 2.0 Core Elements.

In our work we refer to BPMN 2.0, an Object Management Group standard [4]. It is the most used language by domain experts due to its intuitive graphical notation. We have mainly used process diagrams, focusing on the point of view of system users. The following BPMN 2.0 elements (Fig. 1) are the core elements of the language and those we will use on the approach.

- **Events**, which are used to reppresent something that can happen. An Event can be a *Start Event*, representing the point in which the BP starts, while an *End Event* is raised when the BP terminates. Events are drawn as circles.
- **Activities**, which are used to represent a generic work to perform within a BP. An Activity can be atomic - *Task* - or not - *Sub-Process*. Activities are drawn as rectangles with rounded corners.
- **Gateways**, which are used to manage the flow of BP both for parallel activities and choices. Different types of gateways are available, the most used are reported in the following. A *Parallel Gateway* has to wait all its input flows to start and then all the output paths are started in parallel; it can behave as a fork respects to output paths or as a merge respects to input paths. An *Exclusive Gateway* gives the possibility to describe choices both in input and output, it is activated each time the gateway is reached and, when executed, it activates exactly one output path. An *Inclusive Gateway* gives the possibility to select among multiple output paths each time they are reached, it can behave also as inclusive merge. Gateways are drawn as diamonds.
- **Data Objects**, which permit to model documents, data, and other artifact used and updated during the BP, in most of the cases activity take data objects in input and give them back in output. Data objects can also be characterized by a state. A Data Object is represented by a portrait-oriented rectangle that has its upper-right corner folded over. States are represented using text within squared brackets located under the object name.

In order to model variability we combined the BPMN 2.0 standard with an approach based on features modeling, that is illustrated in the following section.

### 2.2 Software Product Line and Feature Modeling

Feature Model (FM) is a modeling approach emerged in the context of Software Product Lines (SPL) in order to support the development of a variety of products from a common platform. The approach aims at lowering both production costs and time in the development of individual products sharing an overall reference model, while allowing them to differ with respect to specific features in order to serve, e.g., different markets [5]. FMs are suitable to represent a family of software products, nevertheless in the last years they have been used also to represent commonality and variability in *Business Information Systems*, introducing the concept of BP family.

In a basic FM a tree representation is used to express relationships among features. Features constraints can be: *mandatory* when characteristics have to be available in each product (Fig. 2-A), *optional* when characteristics can be available or not in each product (Fig. 2-B), and *alternative* when characteristics cannot be present together in the same product (Fig. 2-D) [6].

Basic FM models are too restrictive to represent all the relationships between features which are useful to characterize a family of products [7]. As a result several FM extensions are currently available (e.g. to define feature cardinality).

Adopting richer notations it is possible to include: *OR features* used to express that at least one feature in a set must be included in a product (Fig. 2-C), *include relationship* used to express that a feature selection implies the selection of another feature, this can be a mono or bidirezional relationship (Fig. 2-E), and *exclude relationship* used to express that a feature selection implies to discard another one that is in another part of the tree (Fig. 2-F). As it is shown in the next sections we adopted a enricher FM notation to model BPs variability.

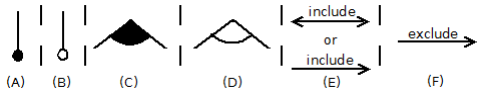


**Fig. 2.** Feature Models Constrains.

## 3 Bike Sharing Systems: a Smart City Scenario

BSS has been chosen as application scenario of our work since it can be easily understood, and at the same time it is sufficiently complex to show the potentiality and needs of process based flexibility. Nonetheless it represents an interesting smart city system where a BSS is typically integrated with other "city functionalities". Within a smart city one of the main objective is to promote the dynamic combination of public service functionalities. New public services are created through pooling and sharing of resources, data and most importantly BP. This results in the need of managing flexibility starting from relevant and common characteristics of a BP.

Around the world the number of cities that have deployed, or plan to deploy, a BSS is continuously raising. Even though at a first glance BSSs could appear rather fixed, they present many variability points. The Oliver O'Brien portal[1] reports, at the time of writing, 104 cities around the world deploying a similar BSS for a total of 9.895 docking stations (i.e. points in the city where bikes can be taken and are generally returned). Nevertheless the characteristics of the listed cities are quite heterogeneous. As it can be imagined, completely different cities will employ variants of a BSS, where variability can relate both to structural characteristics, and to the modality used to provide the functionality to the users. Therefore even though all BSS look similar to each other, in reality the BSS are implemented in many different ways mainly in relation to the requirements of the smart city in which they are deployed. For instance, in case the city is mainly a touristic destination, it is important that the authentication mechanisms are simple and quick, since tourists will not like to waste their time in complex registration procedures. On the other hand, in case the city is not a major touristic destination registration mechanisms can be integrated with a general

[1] http://bikes.oobrien.com/global.php

identity mechanisms, so that the identity card of the citizen can be used to access to the many services made available by the city. Obviously in some case more than one authentication mechanism is necessary.

A BSS typically embeds many different BPs. In particular a BSS will include BP families to support the rental and usage of a bike (called *Bike Travel*), the technical maintenance of bikes (called *Bike Maintenance*), the monitoring of the bikes position (called *All Bike Now*), the user registration (called *User Registration*), the user un-registration (called *User Un-Registration*), and finally to support the users while they are using the bike (called *User Assistance Request*). Each different set include BPs that can be implemented according to many different variants depending for instance on the physical device that are incldued in the concrete system. The following basic architectural requirements/variants can be identified and than used in term of modelling.

**Req1** Most BSSs provide users registration mechanisms to access the service. BSS registration can be done on the fly, using a docking station located near the bike and credit card payment method, or via dedicated form resulting in the delivery of pre-paid smart card (generally called bike card).

**Req2** All BSSs provide a way to unlock bikes from the bike station, take the bike and travel. Existing BSSs implement two different locking mechanisms: (i) bikes can be unlocked using a dedicated device (i.e. smart card or credit card) and (ii) bikes can be unlocked automatically calling a number or asking via Short Message Service for the unlocked code.

**Req3** Some BSSs provide tracking mechanisms to know the position of the bikes in the city, this can be done via Global Positioning System (GPS) or Radio Frequency IDentification (RFID) technologies; no BSS support both.

**Req4** Some BBSs provide rewarding mechanisms that contribute to facilitate the distribution of bikes in different docking station.

**Req5** All the BSSs are maintained thanks to specific agreements with supporting staff to repair bikes when they are at docking station.

**Req6** Few BSSs are maintained thanks to specific agreements with supporting staff to repair bikes in the city during users travel.

**Req7** Some BSSs provide to BSS staff the service "all bike now". It shows the position of all the bikes in the city, eventually in a map.

## 4 Variability Modeling Approach

*FM Extension.* In order to model variability of BPs for the smart city scenario we propose an extended version of FM, named business process Feature Model (bpFM), and then a set of mapping rules from bpFM to BPMN 2.0 fragments, permitting the derivation of a BP skeleton, according to a specific feature selection (configuration) from which a detailed BP can be successively defined. In bpFM features are BP activities and feature constraints express if an activity must or can be inserted in a BP variant, and if it must or can be included within an execution path. It is also possible to add information concerning the input

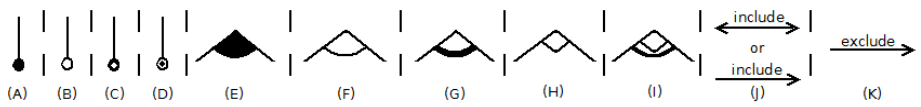


**Fig. 3.** bpFM Constraints.

and output data object related to an activity (feature in the model). In bpFM we include the following constraints.

- *Mandatory Activity* means that the activity must be inserted in each BP model variant and it has to be included in each execution path (Fig. 3-A).
- *Optional Activity* means that the activity can be inserted (or not) in each BP model variant and it could be included (or not) in each execution path (Fig. 3-B).
- *Domain Activity* means that the activity must be inserted in each BP model variant but it could be included (or not) in each execution path (Fig. 3-C).
- *Special Case Activity* means that the activity can be inserted (or not) in each BP model variant, when it is inserted it has to be included in each execution path (Fig. 3-D).
- *Inclusive Multi Activities* means that at least one of the activities must be inserted in each BP model variant, and at least one of them have to be included in each execution path (Fig. 3-E).
- *One Optional Activity* means that exactly one of the activities has to be inserted in each BP model variant, and it could be included (or not) in each execution path (Fig. 3-F).
- *One Selection Activity* means that exactly one of the activities has to be inserted in each BP model variant, and it has to be included in each execution path (Fig. 3-G).
- *XOR Activities* means that all the activities must be inserted in each BP model variant, and exactly one of them has to be included in each execution path (Fig. 3-H).
- *XOR Selection Activities* means that at least one of the activities has to be inserted in each BP model variant, and exactly one of them has to be included in each execution path (Fig. 3-I).

To express input and output data object related to activities, bpFM use the BPMN 2.0 graphical representation including the possibility to associate states to data objects. Incoming and outgoing arrows are used to represent the input and output of the data object from activities. Furthermore our modelling approach supports the guidelines defined in [8] where a semantic for data object is provided. When two different data objects are input or output of the same activity, this means that both are needed or generated to process the activity. A data object cannot be in two different states at the same time. If the same object is linked to the same activity with two different states, it means that the activity execution needs the data object in one of the available states.

*Mapping Rules from bpFM to BPMN 2.0.* According to the feature constraints defined above we have defined a set of mapping rules that permit to automatically derive a BP fragments once a set of features is selected for configuration. For the sake of space we report the textual description of the mapping rules. In Fig. 4 we report the mapping for the case of *Inclusive Multi-Activities* and *XOR Selection Activities*.

- *Mandatory Activity* is always selected and it is mapped as an activity included in the execution path.
- *Optional Activity* if selected it is mapped as a combination of an activity and a gateway condition, so that two execution paths are possible, one including the activity and the other one not. When it is not selected it results with no mapping.
- *Domain Activity* is mapped as a combination of an activity and an exclusive gateway condition, so that two execution paths are supported one including the activity and the other one not.
- *Special Case Activity* is mapped as an activity but differently from the Mandatory Activity could not be selected, in this case no mapping is given.
- *Inclusive Multi Activities* are mapped as combination of an activities and an inclusive gateway condition, so that multiple paths are supported considering selection. In case only one activity is selected it is mapped as an activity (Fig. 4-A).
- *One Optional Activity* is mapped as an activity and an exclusive gateway condition, so that two paths are supported, one including the activity and the other one not.
- *One Selection Activity* is mapped as an activity.
- *XOR Activities* are mapped as a combination of the activities and an exclusive gateway condition, so that alternative paths are supported.
- *XOR Selection Activities* are mapped as activities and an exclusive gateway condition, so that alternative paths are supported; in case only one activity is selected it is mapped as an activity (Fig. 4-B).

The mapping we have defined is also influenced by the presence of data objects. Activities will keep the data object relation considering the state if available also in the generated fragments (Fig. 5).

*Using the Approach.* To model a BP from bpFM the designers follow three main steps. The first step foresees the features selection in the bpFM model to be included in the BP variant. At this stage the designer chooses the features representing the activity she considers necessary to reach the objectives pursued by a BP family. Selected activities, and their relationships, result in a configuration (activities in grey in the bpFM figures) this is the input for the second step where BP fragments are automatically generated thanks to the mapping rules we defined. In particular, for each configuration BP fragments are generated. Fragments can also be already partially ordered depending from the fact that data object dependencies are present in the derived fragments. The final step

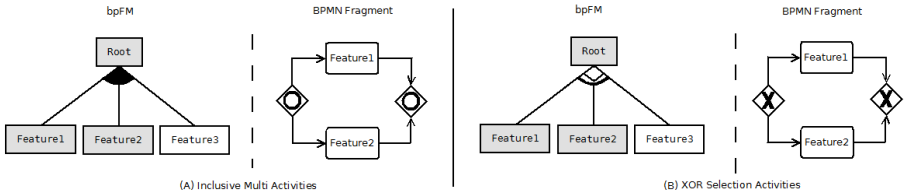


**Fig. 4.** Examples of Mapping Rules

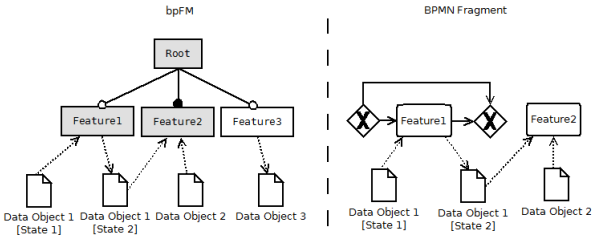


**Fig. 5.** Mapping Data Object.

concerns the modelling of BP variants. At this stage the designer will add control flow among the generated BP fragments. It is worth mentioning that the definition of different control flows, result in different BP variants starting from the same set of generated fragments.

## 5 Modeling Bike Sharing System Supported Variants

According to the application of the proposed approach to the BSS many different BPs families can be defined. Each BP can be implemented differently taking into account the city characteristics. For each BP family we have defined a bpFM model and then according to the configurations we derived BP fragments and related BP variants.

- **Bike Travel** - This is the BP family in which a user (citizen or tourist) registers and accesses a BSS to pick up a bike from a bike station, then she uses the bike to go around the city, and at the end she will return the bike to the same or different bike stations.
- **Bike Maintenance** - This is the BP family in which a bike repairer (staff assigned to the maintenance of the BSS as a whole) checks the components of the bike in order to find damages. In case are damages she has to repair the bike.
- **All Bike Now** - This is the BP family that allow the BSS administrator to retrieve real time positions of the bikes.
- **User Registration** - This is the BP family in which a user (citizen or tourist) registers himself to the BSS in order to use a bike (this BP is optional because registration mechanisms are not always needed).

– **User Un-Registration** - This is the BP family in which a user (citizen or tourist) aims to unregister himself from the BSS (this BP is optional because registration mechanisms are not always needed).
– **User Assistance Request** - This is the BP family in which a user (citizen or tourist) asks for assistance in case the bike breaks during the travel.

For the sake of space we provide here some details about the application of the approach just to the Bike Travel BP family. Fig. 6 reports the bpFM model derived in cooperation with domain experts for the Bike Travel BP family (boxes in grey are those selected in a specific configuration, see below). The users registration functionality is included in the bpFM model using the feature called *Registration*. This is specified as an *Optional Activity*, to mean that the registration is considered optional in the Bike Travel BP since the activity can be inserted (or not) in one BP model variant. The consequence is that the user registration can be available or not in different instances of the BSS, so that in some systems registration has to be completed before taking a bike, whereas in other instances the user does not need to register. The activity (feature) could also be included within an execution path, in particular, if users are not already registered, the activity has to be performed, otherwise it can be skipped. Moreover, when registration is an available feature, it can be done in two different ways represented by the *Credit Card Registration* and *Bike Card Registration* features. This features are linked to the *Registration* feature via a *XOR Selection Activities* constraint. Therefore, if registration is chosen in a configuration, at least one of this two features has to be available, nonetheless users can choose only one mechanism to register to the system. The model is then completed with information concerning the other activities foreseen by the Bike Travel family to represent the system access, the usage of the bike, the bike tracking and the rewarding functionality.

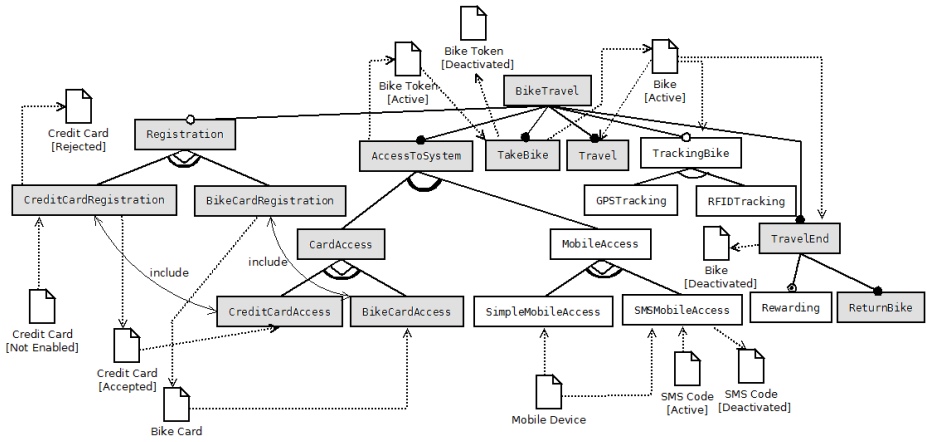


**Fig. 6.** Features selection in the BSS bike travel bpFM

In addition to the modelling of the activities and their relations it is also possible to include in the bpFM model information about data and their usage. In the Bike Travel BP family the following data have been modelled.

- *Credit Card* is related to user registration and system access. States are: *Not Enabled* when registration has not been done, *Accepted* when registration has been successfully done, and *Rejected* when registration has failed.
- *Bike Card* is related to the registration and it is used to access the BSS.
- *Mobile Device* is used to access the BSS.
- *SMS Code* is used together with the mobile device to access the BSS. States are: *Active* before the access has been performed and *Deactivated* as soon as the system is accessed.
- *Bike Token* is used to take the Bike. States are: *Active* before the bike unlock and *Deactivated* as soon as the bike is taken.
- *Bike* is the representation of the bike in the BSS. States are: *Active* after the bike unlock, and in this case we can use the *Bike* related data to know its position, and *Deactivated* as soon as the bike is returned.

Starting from Bike Travel bpFM model we can generate a large set of bike traveling BPs. For instance, in case the designer includes activities to register and to access the system, BPs can be based on Credit Card and/or Smart Card. This configuration results in the selection of the features as represented in the bpFM model of Fig. 6. Such selection are in gray in the figure. Using the defined mapping rules, considering the given configuration, a set of BP fragments are generated. The designer can then derive a fully defined BP variants, for instance (i.e. the one shown in Fig. 7). Similar activities have been carried out to derive

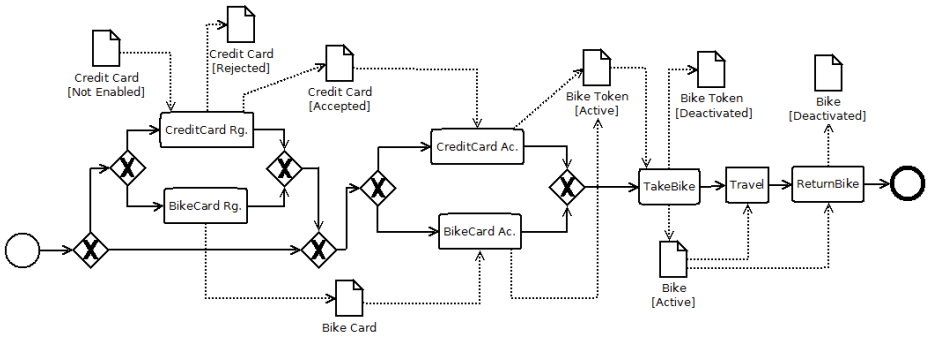


**Fig. 7.** A Bike Travel BP variant.

models for the other BP families so to fully define the Bike Sharing System and its possible variants. The modelling experience resulted to be useful to express BPs variability in real cases studies such as the BSSs. The main advantage is that each BP family can be modelled starting from the bpFM model reducing the complexity of the modelling activity. Moreover, the explicit representation of

variability points permits to improve the understanding scenario. In particular the proposed approach permits to BP managers to have a global view of all the possible BP variants. In addition the approach supports the representation of those data objects considered an important aspect to define needed BP variants. Finally, the variant, automatically derived from BP fragments, permits to introduce a further level of customizability which is particularly useful in order to better fit to different smart city scenarios.

## 6 Related Works

BP flexibility is a hot topic that has been broadly considered during the last years [9] [10]. Considering a smart city context, the most interesting contribution has been given in defining multi perspective BP models variability focusing on BP, and representing separately people and things [11].

With reference to a more static scenario, when an administration has to modernise internal organization structures mainly address organizational changes and we realize that BPM starts to be a suitable tool in order to support innovation strategies. Several initiatives can be reported in such direction in Germany, in Switzerland and in Italy. Worth to be metioned are the eCH framework[2], where a standardization for public service based on BP concepts has been given, and the PRODE project[3] where a reference framework for documents dematerialization involved in service delivery was given. Unfortunately, such contributions aim to create model reference for public sector without considering a way to change BP according to changing legislation, customers needs and environment changes. Other contributions aim to face such needs from different point of views and applying different techniques [12] [13].

Finally, one of the most closed approach to model variability in administration is the configurable BP mode presented in [14]. The authors recognize that many BP in administration are mainly driven by legislation, but they also address some level of freedom regarding BP implementation in different municipalities considering different level of IT system implementation or size. For each BP the authors identified variations among municipalities and integrated them into a single, configurable process model, which can be also executed.

## 7 Conclusions and Future Work

Variability is an aspect that needs to be more and more taken into account when defining systems in changing and evolving context. This is certainly the case for systems to be deployed within a smart city ecosystem. This paper presented an approach to model variability of BP permitting to define in a single model many different variants of the same BP. The defined model can be successively instantiated into a concrete process taking into account the specific characteristics of

[2] http://www.ech.ch/
[3] http://www.progettoprode.it/

the running context. Adopting the approach it is possible to reduce the complexity of managing many different variants of a BP and to share experiences between different smart city initiatives. We plan to extend our modeling effort to other systems and to study the relations and influences among different systems available within the same smart city. At the same time we plan to augment the modeling notation with non functional aspects in order to permit their usage in planning the needed effort to deploy a system in a smart city context. Finally, we intend to investigate the adoption of the notation to drive run-time adaptation, in particular exploring the use of fragment in knowledge BP where a clear BP control flow is not given a priori.

**Acknowledgements** This research has been partially founded by EU project LearnPAd GA: 619583 and by the Project MIUR PRIN CINA - 2010LHT4KM.

## References

1. Chourabi, H., Nam, T., Walker, S., Gil-Garcia, J., Mellouli, S., Nahon, K., Pardo, T., Scholl, H.J.: Understanding smart cities: An integrative framework. In: 45th HICSS. (2012) 2289–2297
2. Weske, M.: Business process management concepts, languages, architectures. 1 edn. Springer (2007)
3. Lindsay, A., Downs, D., Lunn, K.: Business processes–attempts to find a definition. Information & Software Technology **45**(15) (2003) 1015–1019
4. OMG, O.M.G.: (Business Process Model And Notation) BPMN 2.0.
5. Pohl, K., Böckle, G., Van Der Linden, F.: Software product line engineering. Volume 10. Springer (2005)
6. Kang, K.C., Cohen, S.G., Hess, J.A., Novak, W.E., Peterson, A.S.: Feature-oriented domain analysis (foda) feasibility study. Technical report, DTIC Document (1990)
7. Capilla, R., Bosch, J., Kang, K.C.: Systems and Software Variability Management. Springer (2013)
8. Awad, A., Decker, G., Lohmann, N.: Diagnosing and repairing data anomalies in process models. In: BPM Workshops, Springer (2010) 5–16
9. Valença, G., Alves, C., Alves, V., Niu, N.: A systematic mapping study on business process variability. International Journal of Computer Science & Information Technology **5**(1) (2013)
10. Cognini, R., Corradini, F., Gnesi, S., Polini, A., Re, B.: Research challenges in Business Process Adaptability. In: SATTA@SAC. (2014)
11. Murguzur, A., Carlos, X.D., Trujillo, S., Sagardui, G.: On the support of multi-perspective process models variability for smart environments. In: MODELSWARD. (2014)
12. Schminck, A., Eid-Sabbagh, R.H., Weske, M.: egovernment process knowledge ontology - business process knowledge interdependencies in the public administration. In Horbach, M., ed.: GI-Jahrestagung. Volume 220 of LNI., GI (2013) 722–735
13. Gong, Y., Janssen, M., Overbeek, S., Zuurmond, A.: Enabling flexible processes by eca orchestration architecture. In: Proceedings of the 3rd International Conference on Theory and Practice of Electronic Governance. ICEGOV '09, New York, NY, USA, ACM (2009) 19–26
14. Gottschalk, F., Wagemakers, T.A., Jansen-Vullers, M.H., van der Aalst, W.M., La Rosa, M.: Configurable process models: Experiences from a municipality case study. In: Advanced Information Systems Engineering, Springer (2009) 486–500