Information Systems Management, 27:293–308, 2010

ISSN: 1058-0530 print / 1934-8703 online
DOI: 10.1080/10580530.2010.514164

Business Processes Verification for e-Government Service Delivery

Flavio Corradini, Andrea Polini, Alberto Polzonetti, and Barbara Re
Computer Science Division, School of Science and Technologies, University of Camerino, Camerino (MC), Italy

Domain experts knowledge represents a major source of information in the design and the development of user-centric and distributed service-based applications, such as those of e-government. Issues related both to the communication among domain and IT experts, and to the implementation of domain dependent requirements in service-based applications, have to be carefully considered to support both Public Administrations efficiency and citizen satisfaction. In this article, we provide as user-friendly approach toward business process assessment via formal verification. Starting from a semi-formal notation, well understood and largely used by domain experts, we provide a mapping to a formal specification in the form of a process algebra. This transformation makes possible formal and automatic verification of desired quality requirements. The approach has been already applied, with encouraging results, in the e-government domain to verify the quality of business processes related to the delivery of e-government digital services to citizens. Moreover, the approach is supported by a plug-in for the Eclipse platform permitting to have an integrated environment in which to design the process model and to assess its quality.

Keywords quality; e-government; business process; formal verification

1. INTRODUCTION

The advent of the Internet in the 90s supported government ambition to provide better services to citizens through the development of ICT-based solutions. Thanks to the Lisbon conference in 2000, which covered and investigated this topic, e-government has been recognized as one of the major priorities in the innovation process for Public Administration (PA).

The current maturity of ICT technologies enables a relatively simple development and delivery of e-Government Digital Services (GDSs). European Commission studies show that 83% of basic GDSs are today available on-line (Colclough & Tinholt, 2009). Comparing the same data with the ones reported in 2006 and 2007, by similar surveys (Colclough, 2007, Wauters & Colclough, 2006), it is evident that the interest of the PA in GDSs is quite strong and quickly rising. Nevertheless, a different perception is evident if e-government effectiveness, i.e., real usage by citizens is considered. In this case, the result is that many of the available GDSs are seldom accessed and used by citizens. In (Bavec, 2008) the authors recognize that “practical experiences and researches confirm that users’ acceptance is not guaranteed per se. Public approval is quite often below what developer expected.” In other words, citizens do not use e-government services just because they are available. Many different reasons contribute to the highlighted situation, and certainly, the well-known digital divide phenomenon (Norris, 2001) can be considered as one of the possible causes of low GDSs usage. Nevertheless our impression, corroborated by an informal investigation among the people working in our department, so with high ICT skills, is that other important factors strongly contribute to the underlined scenario.

In our view, important factors that can justify low GDS usage, can be reconducted to specific properties of the processes governing the various steps of GDSs delivery. We think that service delivery strategies are often too focused on technological aspects skipping social, anthropological, and organizational views. Moreover, service delivery processes too often reflect already available processes, typically derived and conceived in a “human-only based” interaction setting. Thus, in order to define a good GDSs delivery process, it is certainly important to take advantage of new opportunities provided by the availability of ICT technologies (Dhillon & Mishra, 2007), nevertheless, social, anthropological, and organizational characteristics should not be forgotten. So, for instance, in a “human-only based” process, citizen trust is raised by direct interactions with the civil servant, which represents a citizens direct access point to information concerning the request and its execution status. When delivering a GDS, it is important to keep in mind that access to a web site does not provide the same level of trust per se. Too many things remain hidden in the citizen’s perception. It is important, therefore, to revise the process to introduce mechanisms that can help to increase citizens trust. For instance an insurance company is currently advertising in Italy as a new way of delivering the customer care service.

Address correspondence to Barbara Re, Computer Science Division, School of Science and Technologies, University of Camerino, Via Madonna delle Carceri, 9, 62032 Camerino (MC), Italy. E-mail: barbara.re@unicom.it

In particular, in order to increase trust perception, the company decided to redirect all the incoming calls from a given customer care support service. In order to increase trust perception, the company decided to redirect all the incoming calls from a given customer always to the some phone operator. The idea is clearly to create a more trustworthy connection among the customers and the organization. We think that similar considerations should be taken into account when planning GDSs delivery.

In this article, we intend to introduce a methodology and a tool to formally and automatically assess the quality of a designed Business Process (BP) with respect to defined quality requirements. As detailed in the following, the proposed approach foresees that a domain expert will design a process to fulfill specific goals. At the same time, domain experts will describe quality requirements that a BP applied in the given domain should fulfill. The business process and the requirements are transformed—through specific mappings we have defined— into formal specification in order to assess if the BP actually satisfies the required properties or not. In case some properties are violated, the BP designer will be asked to review the process model. The assessment step is based on well-established techniques in the area of formal verification. The approach has been codified in a plug-in for the Eclipse platform resulting in an integrated environment for BP specification and verification.

The approach has been also applied to the evaluation of processes in the GDS delivery for which we has previously defined a quality framework (Corradini, Hinkelmann, Polini, Polzonetti, & Re, 2009b). In particular, the defined framework provides a five-dimension evaluation schema to assess several process delivery related requirements that influence citizens perceptions and their usage of a GDS. For each dimension, we also provide design guidelines to GDS developers that should structure the delivery process according to corresponding feedbacks, in order to derive a highly usable and used service.

To motivate and validate our discussion, we report some evaluations on the current Italian scenario. Finally, it is worth noting that in line with what is recommended in (Salem, 2007), the framework can be a first tool for practical benchmarking of PA GDSs delivery process facing some important issues still unexplored by the current benchmarking studies (Berntzen & Olsen, 2009).

In the next section, we provide some background material to make clearer both the motivations of our work, and the techniques we investigated. In Section 3, we describe the formal mappings we have defined, and the phases composing the formal verification approach we propose. Successively, Section 4 describes a tool-set we provide to make the application of the approach easier. A real experience is then reported in Section 5, followed by a Related Works Section. Finally, we draw some conclusions and lines for future investigations in Section 7.

2. BACKGROUND

2.1 Business Processes and BPMN

We refer to a business process as a collection of related and structured activities undertaken by one or more organizations in order to pursue some particular goal (Lindsay, Downs, & Lunn, 2003). Within an organization, a BP results in the provisioning of services, or in the production of goods, for internal or external stakeholders. Moreover, BPs are often interrelated since the execution of a BP often results in the activation of related BPs within the same or other organizations.

Business Process Management (BPM) supports BP experts providing methods, techniques, and software to model, implement, execute, and optimize BPs which involve humans, software applications, documents and other sources of information (Harmon, 2004).

Recent works show that BP modeling has been identified as a fundamental phase in BPM. The quality of BPs resulting from the BP modeling phase is critical for the success of an organization. However, modeling BPs is a time-consuming and error-prone activity. Therefore, techniques that help organizations to implement high-quality BPs, and to increase process modeling efficiency has become an highly attractive topic both for industries and for the academy. Certainly, many different commercial tools have been developed to support BPM; nevertheless, for the modeling phase, they mainly provide support for BP editing and syntactical analysis. To the best of our knowledge, none of them introduces and correspondingly support formal verification techniques.

Different classes of languages to express BPs have been investigated and defined. There are general purpose and standardized languages, such as the Business Process Modeling Notation (BPMN) (White & Miers, 2008), the Event-Driven Process Chain (Mendling, 2008), and the UML Activity Diagrams. There are also more academic related languages, with the Yet Another Work-flow Language (Van der Aalst & ter Hofstede, 2005) based on Petri Nets as probably the most prominent example. Among the listed languages, differences are related to the level of rigor, going from semi-formal languages, with a precise syntax but with semantic given in natural language, to formal languages for which the semantic is provided mapping the various constructs to well founded mathematical theories.

In our work, we refer to BPMN 1.1, in the following just BPMN. This is certainly the most used language in practical context also given its intuitive graphical notation. Nevertheless, BPMN does not have a clearly defined semantic. For this reason, and in order to permit formal verification of Business Processes, we defined a mapping of BPMN constructs to CSP processes.

A BPMN process is made up of BPMN elements composed to express the desired sequence of activities. Figure 1 provides an overview of BPMN elements related to control-flow specification. In particular, a BPMN process can include objects,

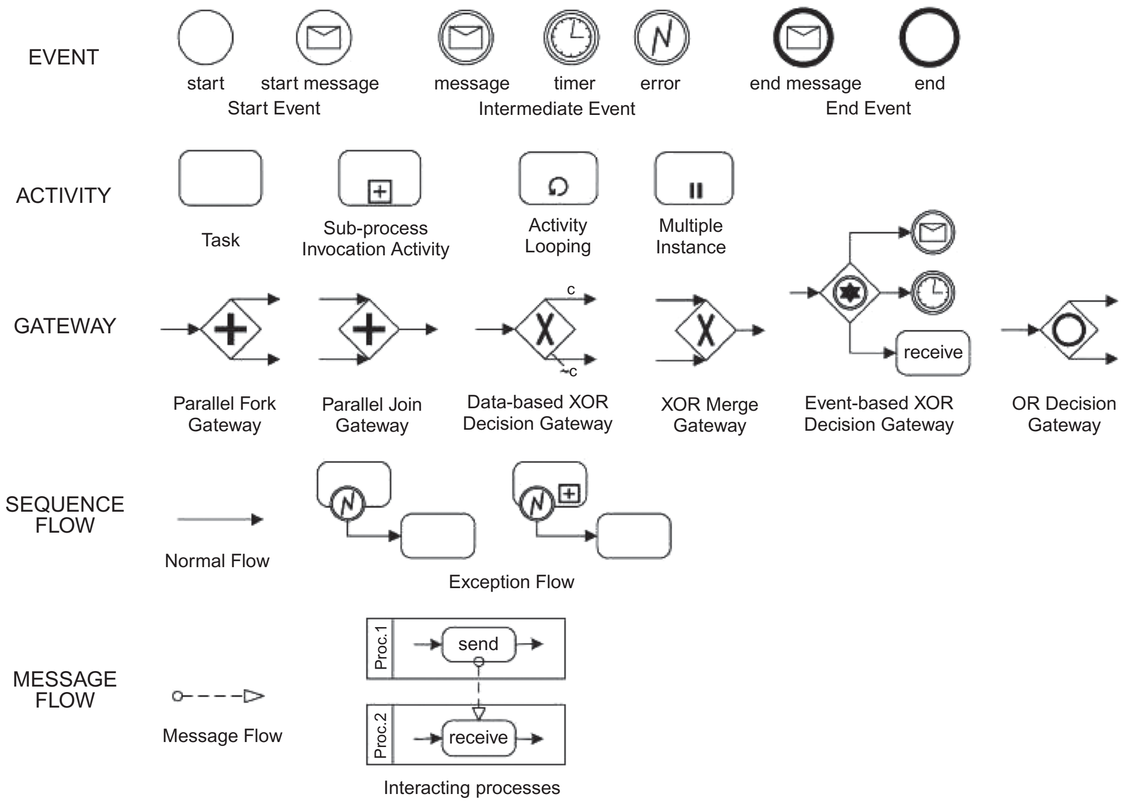


FIG. 1. BPMN elements.

sequence flows, and message flows. An object can be an event, an activity, or a gateway. Events are necessary to start, terminate, or interrupt the defined process. An activity can be an atomic task or a composed activity including other atomic activities. A gateway to influences the path followed by a process, and for this, they are also referred as routing constructs. A sequence flow links two objects in a process model and denotes a control flow (i.e., ordering) relation. Finally, message flows are used to capture the interactions among processes.

Being an Object Management Group (OMG) standard, BPMN is constantly undergoing revisions and extensions. The next release will be BPMN 2.0 that is currently in the "Request For Proposal" status since June 2007. The final release is currently planned for 2011. BPMN Version 2.0 solves most of the problems of BPMN, in particular it has a more structured and precise meta-model. Nevertheless, a formal semantics is still missing, so that the basic principles and motivations of our approach will be still valid after its final release.

2.2 Process Algebras and CSP

Process algebras are used to formally model concurrent and communicating systems, and to provide a tool for the high-level description of interactions, communications, and synchronization among a collection of independent processes (Bergstra & Klop, 1982; Baeten, 2005). They also define composition operations that allow process descriptions to be manipulated and integrated. Moreover, the resulting formal representation of the system is suitable to be submitted to formal reasoning techniques to assess, if the system satisfies given properties or not. Leading examples of process calculi include CSP, CCS, ACP, LOTOS, π-calculus, ambient calculus, PEPA, and many others.

CSP is an event based notation primarily aimed at describing the sequencing of activities within a process, and the synchronization (or communication) between different processes. Events represent a form of cooperative synchronization between the processes and the environment. Both the processes and the environment may influence the behavior of the other by enabling or refusing certain events or sequences of events.

In CSP, a process is a pattern of behavior and a behavior consists of events, which are atomic and synchronous between the environment and the process. Events can be constructed using the dot operator '.' to form compound events; often these kind of events are used to implement channels permitting to represent a more structured communication schema among processes. Below, we report the grammar in BNF for the CSP language, where P and Q represent generic processes and e represents a generic event.

$$P ::= STOP|SKIP|e \rightarrow P|P\Box Q|P||_A Q|P;Q \quad e ::= x|x.e \quad (1)$$

In detail, the process *STOP* represents a deadlocked process, where the process *SKIP* is used to represent a successful termination. Process e→*P* denotes a process able to perform the event e, after which it will behave as process *P*. Process *P*□*Q* refers to the external choice between processes *P* and *Q* The process is ready to behave either as *P* or as *Q* and external factors will make the choice among this two possibilities. Process *P*||*Q* denotes the interleaved parallel composition of processes *P* and *Q*. Process $P||_A$ denotes the partial interleaving of processes *P* and *Q* which share events listed in the event set *A*. Process *P*;*Q* denotes a process ready to behave as *P* and after that *P* will successfully terminate, the process will behave as *Q*. Finally, events can be composed using the dot operator ".". Compound events can be used as channels to communicate data objects synchronously between the process and the environment. CSP processes are closed terms built up out actions and other processes using some operators. The original CSP process algebra includes other operators that we do not present here since they are not strictly useful for the purpose of our presentation.

Finally, the operational semantics of CSP is typically given by a set of inference rules defining a mapping from CSP terms to transition systems. We do not report such rules here and the interested reader can refer to (Hoare, 2004).

2.3 Formal Verification

In the context of software systems, formal verification is the act of proving or disproving the correctness of a system, with respect to given formal specifications or properties, using methods based on sound mathematical tools. Many different formal approaches can be applied to systems verification. Our interest is mainly in model checking techniques (Clarke, Corumberg, & Peled, 2000), which propose to systematically explore (and when possible explore exhaustively) an operational model to verify if it satisfies a set of given properties or not. Implementation techniques include state space exploration, symbolic state space enumeration, abstract interpretation, symbolic simulation, abstraction refinement, and others. The properties to be verified are described as goals to reach or conditions that systems states have to satisfy. Reachability analysis, deadlock-free analysis, and generic temporal logic properties, such as those expressed using Linear Temporal Logic (Emerson, 1990), are typical properties that can be verified on a complex system.

Since its first inception many model checking tools have been proposed and developed with the definition of several formalisms to represent the system, and languages to specify the properties to verify. We refer to SPIN (Holzmann, 1997), SMV (Carnegie Mellon University—School of Computer Science, 1998), and Pat (Sun, Liu, & Dong, 2008), just to cite a few. In our work, we integrate the PAT model checker due to its flexibility and since it uses the CSP formalism as input language.

3. FROM BP SPECIFICATION TO BP CHECKING

In this section, we outline the elements composing the BP formal verification approach we have defined. The use of formal mechanisms to verify properties of complex BPs has been already advocated by other authors [see for instance (Wynn, Verbeek, van der Aalst, ter Hofstede, & Edmond, 2009)]. Our work aims at providing to BP and domain experts the power of formal verification techniques still allowing the use of graphical notation with which they are already acquainted. The approach, which is sketched in Figure 2, relies on the following three main steps:

i. Business Process and quality requirements specification via a user-friendly notation;
ii. Mapping of a process specification and of a set of quality requirements to a CSP like notation and to a set of goals, respectively;
iii. Formal verification of defined processes with respect to specified set of properties (goals).

If the verification phase ends, and it highlights some problems—i.e., at least one of the properties defined by domain experts results to be violated—the process should be restarted.

The remaining part of this section is structured as follows. Section 3.1. describes the mapping from BPMN to a CSP model, Section 3.2. introduces issues related to requirements specification and finally Section 3.3. details some aspects of the verification.

3.1 BPMN2CSP

The BPMN 1.1 specification does not define a precise semantic for the provided constructs, so one of the objectives of the upcoming version of this specification is to clarify the semantic of BMPN constructs. Nevertheless, even if a huge effort has been spent toward this objective, the draft version still

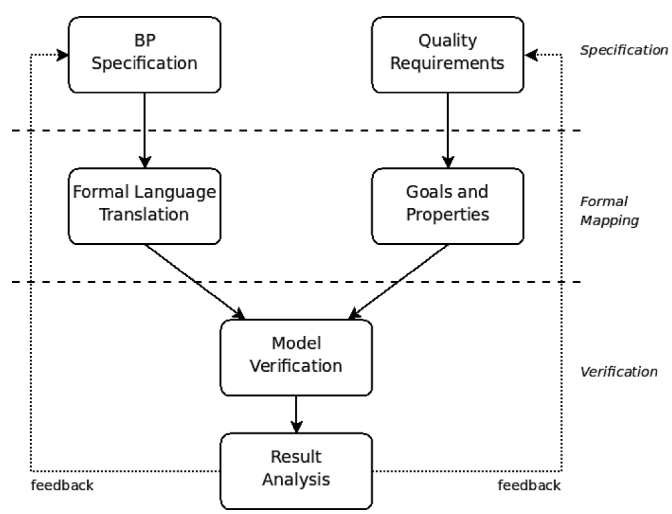


FIG. 2. Proposed approach logical view.

contains descriptions in natural language. On the other hand, the request for comments for BPMN 2.0 is still open, so it is possible that some updates will be included before the final release. In our work, we have defined a precise semantic for BPMN constructs through a mapping to CSP processes. Providing a formal semantic to a semi-formal language results in the definition of a unique interpretation for each construct, which could possibly be different from that intuitively given by some BP developer. We do not think that this is a big issue for our purpose given that the objective is mainly to verify properties. So the BP developer should be in any case alerted by a possible interpretation that could lead to a low quality process. Nevertheless, in order to reduce such a risk, we have derived design rules that impose some restrictions on the usage of the available BPMN constructs. This is particularly important when BPMN constructs miss specifying details for verification purpose, or when they could lead to a particularly ambiguous interpretation.

Our mapping covers all the core BPMN constructs and almost all the constructs introduced by the OMG notation. Few constructs dealing with transactions, such as compensation events, cancel events, or time have been kept outside of our mapping. The main reason for this choice is that they are seldom used in practice (zur Muehlen and Recker, 2008)—at least in the e-government domain. Another reason relates to the fact that time properties cannot be verified within the proposed approach.

In order to apply our approach, and the tool-set we provide, BP developers have to abide by the following constraints.

- Tasks have to be typed to support specific domain-dependent characterizations.
- Tasks can include at most one type of communication (send or receive); they can not include both. In this way, we ask the BP developer to explicitly provide the order in a sequence of messages exchange. Delivery of messages are assumed to happen before exiting from the task where message reception are assumed to happen while entering the task.
- Messages have to be typed to support specific domain-dependent characterizations.
- Pools have to be typed to address the role they play in the process.
- Loops have to be explicitly represented as loop-task or loop-subprocess. No implicit cycles are admitted in the process design. This constraint allows more structured BPs, and thus, avoids the presence of a loop generated by a kind of unconditional jump. Besides making the verification step more difficult, the presence of such loops make the specification more complex and less understandable.
- Collapsed sub-processes are not supported. Moreover, for each sub-process, BPMN end and start events have to be explicitly provided since they support the trigger of elements inside the sub-processes.

When a BP has been modeled according to the constraints listed above, the approach permits one to derive a CSP model (input format for the selected model checking tool in the real implementation of the approach) from the BPMN model. The mapping has been defined according to the following general principles:

- Each BPMN graphical object included within a pool is formally represented by a CSP process or a parallel execution of generated CSP processes—we will name such process *Element CSP*.
- Each pool is mapped to a parallel composition of *Element CSP* processes with a barrier synchronization. In this case no message exchange will be observable—we will name such process *Private CSP*.
- The whole process results from the parallel execution of the *Private CSP* processes including their interactions implemented via messages exchange—we will name such processes *Abstract CSP*.

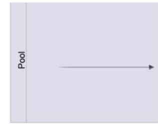
Due to space limitation within Figure 3, we report few mapping rules. All the rules we have defined provide a denotational semantic to the various BPMN elements and to their composition in term of a complex CSP process. A wider discussion on the mapping can be found in (Corradini, Falcioni, Polini, Polzonetti, & Re, 2009a).

- The rule to transform the BPMN pool elements produces a CSP global constant (Figure 3a). So, the general idea is that each participant will be identified by such a constant value.
- The rule to transform the BPMN sequence flow element produces a CSP process (Figure 3b). Such CSP process is able to perform an event `esc` after which it will perform the event `enter`. Both events are characterized by the identifiers of the flow and of the pool that contains the flow itself. The general idea is that the CSP process related flow is started by an interaction with the environment. Then, it is requested the synchronization of the event `esc` with the corresponding `esc` event generated by another BPMN element where such flow is outgoing. After that synchronization with the environment via the event, `enter` is requested. Also in this case, the environment is represented by the CSP process generated by a BPMN element where the same control flow is incoming. When the CSP process output of the mapping from BPMN flow elements returns the BPMN flow is fired and the whole CSP process terminates with success. The CSP process can also terminates in case a BPMN event termination occurs according to the process and sub-process where the flow is placed. A similar mapping has been done for the rule related to the conditional flow.


```
PoolID = "poolID";
PoolName = "poolName";
```

```
#define poolName poolID;
```

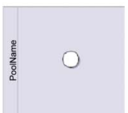
(a) Pool.




```
PoolID = "poolID";
SubProcID = "subProcID";
FlowID = "flowID";
```

```
FlowID(flowID, subProcID, poolID) =
(esc.flowID.poolID → enter.flowID.poolID → Skip; )
□ (terminate.poolID → Skip; )
□ (terminate.poolID.subProcID → Skip; )
```

(b) Flow.




```
PoolID = "poolID";
StartID = "startID";
OutgoingEdges = "flowID_1, flowID_2, ..., flowID_n";
```

```
startID(startID, flowID_1, ..., flowID_n, poolID) =
enable.startID.PoolID → (esc.flow_1.poolID || ... || esc.flow_n.poolID) → Skip;
```

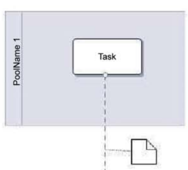
(c) Start Even.


```
PoolID = "poolID";
SubProcID = "subProcID";
TaskID = "taskID";
IncomingEdgesIDs = "flowInID_1, ..., flowInID_n";
OutcomingEdgesIDs = "flowOutID_1, ..., flowOutID_m";
TaskIDTransparancy = "traspValue";
```

```
var TaskIDTransparancy = "traspValue";

TaskID(taskID, flowInID_1, ... , flowInID_n,
flowOutID_1, ... , flowOutID_m, subProcID, poolID) =
((enter.flowInID_1.poolID || ... || enter.flowInID_n.poolID) →
enable.taskID.poolID → work.taskID.poolID →
(esc.flowOutID_1.poolID || ... || esc.flowOutID_m.poolID) → Skip; )
□ (terminate.poolID → Skip; )
□ (terminate.poolID.subProcID → Skip; )
```

(d) Task.

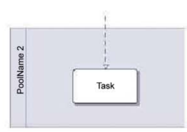



```
PoolID = "poolID";
SubProcID = "subProcID";
TaskID = "taskID";
IncomingEdges = "flowInID_1, ..., flowInID_n";
OutcomingEdges = "flowOutID_1, ..., flowOutID_m";
OutcomingMessages = "msgOutID";
DataObject = "dtObjectName";
chType = "com" or "coll" or "sInt";
CollaborationTypeMsgID = "ColElID";
TransparencyMedMsgID = "TraElID";
SharingDatMsgID = "SharElID";
```

```
Channel chTypemsgOutID 0;

SendTaskID(taskID, flowInID_1, ... , flowInID_n, dtObjectName
flowOutID_1, ... , flowOutID_m, subProcID, poolID) =
((enter.flowInID_1.poolID || ... || enter.flowInID_n.poolID) →
enable.taskID.participantID → work.taskID.participantID →
chTypemsgOutID!dtObjectName → ({CollaborationTypeMsgID = ColElID,
 TransparencyMedMsgID = TraElID, SharingDatMsgID = SharElID} →
(esc.flowOutID_1.poolID || ... || esc.flowOutID_m.poolID) → Skip; )
□ (terminate.poolID → Skip; )
□ (terminate.poolID.subProcID → Skip; )
```

(e) Task sending.




```
PoolID = "poolID";
SubProcID = "subProcID";
TaskID = "taskID";
IncomingEdges = "flowInID_1, ..., flowInID_n";
OutcomingEdges = "flowOutID_1, ..., flowOutID_m";
IncomingMessages = "msgInID";
CreativeTypeMsgID = "CreElID";
ProactiveTypeMsgID = "ProElID";
ReactiveTypeMsgID = "ReElID";
TaskIDTransparancy = "traspValue";
```

```
var TaskIDTransparancy = "traspValue";

ReceiveTaskID(taskID, flowInID_1, ... , flowInID_n,
flowOutID_1, ... , flowOutID_m, subProcID, poolID) =
((enter.flowInID_1.poolID || ... || enter.flowInID_n.poolID)
→ enable.taskID.participantID
→ channelTypemsgInID?msgIn → {CreativeTypeMsgID = CreElID,
 ProactiveTypeMsgID = ProElID, ReactiveTypeMsgID = ReElID}
→ work.taskID.participantID
→ (esc.flowOutID_1.poolID || ... || esc.flowOutID_m.poolID) → Skip; )
□ (terminate.poolID → Skip; )
□ (terminate.poolID.subProcID → Skip; )
```

(f) Task receiving.

FIG. 3. Examples of mapping rules from BPMN to CSP.

- The rule to transform the BPMN start event elements produces a CSP process (Figure 3c). Such CSP process is able to perform the event `enable` after which it will execute a set of `esc` events. The first event is characterized by the identifiers of the BPMN start and of the pool that contains such element. Each `esc` event is characterized by the identifier of the outgoing flow and by the pool identifier that contains such a flow. The general idea is that the CSP process is immediately enabled without any interaction with the environment. Then the synchronization of the events `esc` with the corresponding `esc` event generated by BPMN flow elements is requested. When the CSP start event process returns, the event related to the outgoing flows are fired, and the CSP process is successfully terminated. Also in this case the process can be terminated when a BPMN event termination occurs. A similar mapping rule has been produced for the events typed with conditions. Both the BPMN start and the end event are considered in our mapping. In this case, the general idea is that the CSP process enables the incoming flows on the end event implemented via other CSP processes related to flows BPMN element and then it consumes itself.

- The rule to transform the BPMN simple task elements produces a CSP process (Figure 3d). Such a CSP process is able to perform the `enter` event; after that, it will perform the `enable`, `work`, and `esc` events. The first event is characterized by the identifiers of the incoming BPMN flows and of the pool that contains such elements. The second and the third events are characterized by the identifier of the task and of the pool that contains the task itself. Finally, the fourth event is characterized by the identifier of the outgoing BPMN flows and by the pool that contains such flows. The general idea is that the CSP process firstly interacts with the environment (with the CSP processes related to the incoming flows); secondly, the main task in enabled and executed; and finally, the process implements another interaction with the environment (with the CSP process related to the outgoing flows). More specifically: first, the events `enter` are synchronized with the corresponding set of `enter` events generated by BPMN incoming flows, second, the `enable` and the `work` events are consumed without interacting; and finally, the synchronization of the events `esc` with the corresponding set of `esc` events generated by BPMN outgoing flows elements is requested. A similar behavior is observable for the rules related to tasks characterized with loops, multi-instance both in parallel and sequence and messages (Figure 3e–f). For the concerns of task sending and receiving messages, we introduced a CSP dedicated channel enabling the message exchange.

3.2 Domain Related Quality Requirements and Mapping

Domain-related quality requirements that generally characterize all the process in a given domain should be defined by domain experts. Nevertheless, domain experts may not have enough skills in formal languages; so, in general, they could not be able to describe such requirements using a given formal notation. Here, our contribution is on the codification of such domain knowledge within a tool defining a set of property templates, which should be satisfied by any process in the given domain. Such templates also have a checklist that can be used during the design of any BP.

Certainly the codification of a quality framework, i.e., the derivation of the property templates, requires the collaboration among domain and IT experts. Depending on the domain requirements, on the mapping rules, and on the model checking tools used for the codification of the framework can be exploited according to different approaches. In order to actually verify some of the properties, it is possible that it is necessary to directly intervene on the mapping rules to add statements expressely related to the domain requirements or to verification functionalities. This is what we did in our case. Thus, the derived checklist hides a set of assertions that can be assessed thanks to the addition of global variables within the mapping rules. Each mapping rule influences the verification of a property redefining a global variable, which is successively combined with other global variables to check the whole assertion.

3.3 Verification

The verification phase is based on model checking techniques. Reachability analysis is applied in order to assert whether the goals that are generated from the properties specification are fulfilled or not. The model checker will apply a search algorithm to repeatedly explore unvisited states until a state where the condition is true is found or all the states will have been visited.

It is worth mentioning that for the BPs typically found in the e-government domain, the approach does not suffer from the state explosion phenomenon as will be shown by some experimental data we already collected in Section 5. In very general terms, this problem relates to the huge, possibly infinite, number of states that even a simple model could generate. The result is that when this phenomenon appears an exhaustive exploration of the state space becomes infeasible or too much expensive. A reduction of such a risk can obviously contribute to the definition of the mapping. In our case, one of the choices that we took to mitigate such a risk refers to the fact that for the purpose of reachability analysis the data can be often ignored or mapped to small finite sets. Nevertheless, in different domains, this could not be the case, and the BP developer will have to take into account this possible hurdle to the applicability of this kind of formal verification technique.

4. BP4PA: TOOL CHAINS FOR BUSINESS PROCESS VERIFICATION IN THE E-GOVERNMENT DOMAIN

As underlined in the introduction, one of the main problem in the e-government field is the scarce usage of GDSs. We are aware that different reasons led to such a situation, but we believe that to promote services usage, delivery related processes should be carefully evaluated in line with specific quality requirements. Given the complexity and the heterogeneity of e-government applications, a human only based investigation on BP characteristics is not enough to guarantee a high quality level. Instead, this domain seems to have reached a maturity level in which more systematic and standardized techniques can be deployed. The approach we propose fits such a scenario, and it can improve the effectiveness of e-Government Digital Services. In particular, it provides a toolset for the application of formal verification techniques, such as model checking, to assess the quality of GDS delivery related processes.

The remaining part of this section is structured as follows. Section 4.1. describes our tools chain where Section 4.2. describes the BP4PA framework focusing on issues related to properties specification in the specific domain of the GDS delivery processes.

4.1 Implementation Details for the BP4PA Plug-In

The formal verification approach illustrated in this article is supported by a plug-in available for the Eclipse Framework that can be freely downloaded at the BP4PA web page (http://bp4pa.sourceforge.net/index.html). The plug-in permits to have a fully integrated and user friendly environment which supports domain experts both in the BP specification phase, and in the verification phase. In particular, our plug-in is integrated in an Eclipse extension such as the BPMN modeler, and it uses the functionalities of the PAT model checker (Sun et al., 2008). The CSP model is derived taking advantage of the Eclipse Modeling Framework (EMF), which is a powerful mechanism made available in the Eclipse platform to define meta-models. EMF, together with other framework, enables the graphical rendering of the BPMN constructs, which is at the base of BPMN modeler. Therefore, through EMF, and the API it makes available, it is possible to interact with the defined BPMN model to retrieve the list of elements which have been included within a BPMN specification. In this way, it is possible to implement a simple parser so for each BPMN element will generate the corresponding CSP code; in our case, we use the syntax of the PAT model checker and adhere to the rules defined in Section 3.1. Similarly, the code generation will include the specification of variables enabling the checking of relevant quality attributes as specified in the framework illustrated in the following of this section. After that, the transformation of the BPMN specification to the corresponding CSP model has been carried on, and the verification step can take place. To make this step easier, we integrated the starting of the PAT model checker within the eclipse framework. As a result, the whole tool chain is integrated in a unique IDE. It is worth noting that the current implementation refers to a quality framework included within our tool as illustrated in Figure 4. The framework has been mainly defined by us in collaboration with experts in the domain of BPs for public administrations. Nevertheless, the techniques we used to implement the framework make the extension of the framework with new properties easier.

Figure 4 reports the various components included in the BP4PA tool putting them in relation to the various elements described in Figure 2.

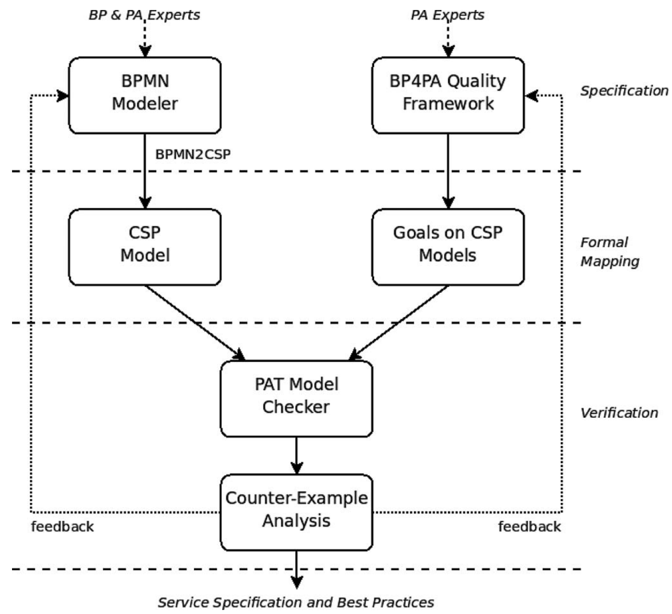


FIG. 4. BP4PA tool chain.

4.2 Specification and Checking Issues

Imagine being in a local administration where promoting citizen inclusion is one of the main goals. e-Government business analysts should continuously review processes to be sure that they are in line with quality requirements. Nevertheless, we believe it is often the case that the employees do not have high skills in ICT methodologies and techniques within these offices, and they will carry on the quality checking mostly by hand. In this context, our approach supports e-government business analysts, with an easy to use BP development environment that includes quality verification. Moreover, we drive analysts in the process specification by providing suggestions and defining constraints for the design phase. For the BPMN modeler concerns in particular, we have introduced a set of constraints, in line with those described in Section 3 that impose some additional rules on the business process modeling phase. As we have noted, these constraints aim at making the verification phase more effective and making the specification less ambiguous. The inclusion of such additional constraints is made possible by the extensible structure of EMF, which permits one to semantically enrich the managed data structures that correspond to the developed models. In particular, such constraints will specify that:

- Every time a message is introduced in the process we both request to set up the message type, and to specify if the message informs the citizen on the availability of additional useful services. The first constraint is needed to check coordination level and the second one is for control level.
- Every pool implementing the role of the citizen has to be typed with user characterization. This is requested to check the control requirement. At the same time,

pool can be typed with the mediator role, to implement process tracing and activity aware transparency.
- It is possible to characterize a pool element as a database. Such a pool will receive messages from other pools/tasks supporting the storage of the information enabling their reuse
- All tasks have to be enriched with the transparency characterization. This is needed to verified the activity aware transparency checking.

With reference to quality requirements modeling, we decided to provide a list of properties, in the form of a checklist, based on the BP4PA framework that has been introduced and illustrated in (Corradini et al., 2009b). The BP4PA framework permits one to specify a different level of quality for characteristics such as those of coordination, control, sharing, transparency, and inclusion. Having defined the quality requirement framework, our effort has been focused on providing a mapping to goals to be satisfied by Business Processes related to e-Government Services Delivery. In the following section, we discuss how the goals are used and checked at design time on BP under development.

The *coordination quality* requirement predicates over the interactions among Public Administrations involved in the delivery process. In particular, with coordination, we mean the capability of two or more Public Administrations to work together in order to accomplish a common goal and trought the usage of ICT technologies. To check coordination levels, we assume that message exchanges are typed explicitly. Three different levels of coordination have been identified: communication, collaboration, and semantic integration. Our verification of coordination characteristics will conclude that the service reach the quality goal linked to the coordination level specified by the user if interactions among the involved administrations are implemented via ICT technologies, and the sequence of messages exchange fits or is higher than the specific coordination goal. In other words, we require interactions among participants, and we check the type of messages exchanged. If different parts of a process satisfy a different type of interactions, the lower level of quality is considered and used to rank the business process. Lack of coordination is observed only in the case of more than one participant contributing to the business process without implementing messages exchange. Figure 5 shows how to check the coordination level on a simple business process using the BP4PA interface. In such a case, two interactions are implemented via messages exchange typed with communication and collaboration levels, respectively. Results of the quality check show that the process fits in the collaboration level respect to the coordination quality requirement.

The *control quality* requirement predicates over the first interaction in the BP among administrations and citizens. With regard to control, we refer to the paradigm applied to drive the GDS delivery from its start to its final fulfillment. In the analysis phase, the resulting control level can be reactive or proactive. Our check on control concludes that the service reaches the quality goal linked to the specific reactive control level if the first message goes from the citizens toward the PA. Otherwise, if the first message is incoming to the citizens pool, the service is recognized as proactive. There is also the case where services are both reactive and proactive (two parallel process are implemented in the citizen pool with separate start and end event), and in this condition, no control is observable. In other words, we check the order in which the interactions among PA that participate to the services delivery and citizens are implemented. We also consider the possibility to implement a creative service distribution. This characteristic refers to the promotion of related, and maybe relevant, services. Creativeness can be provided as an enrichment on the reactive and proactive way of implementing services delivery control. In this case, we check if there is

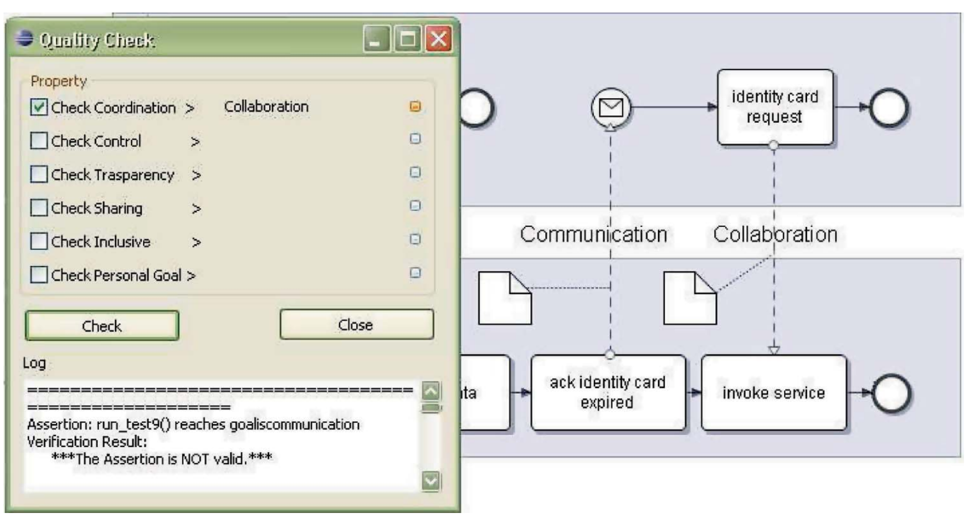


FIG. 5. Coordination check: collaboration.

a message characterized with creative features going from PA to citizens. To provide simple examples on the control level we refer to the cases in Figures 6, 7, and 8. In such cases, we show a proactive, a reactive, and a process that do not satisfy any level of control respectively.

The *sharing quality* requirement predicates over the way in which the PA handles and shares citizen data with other administrations in order to participate in the delivery of a specific GDS. To check sharing levels, we assume that a specific participant typed with knowledge repository role is present in the service delivery. Our verification on sharing concludes that the service reaches the quality goal linked to the specific sharing level if such a role is included in the process and other participants interact with it. In other words, we

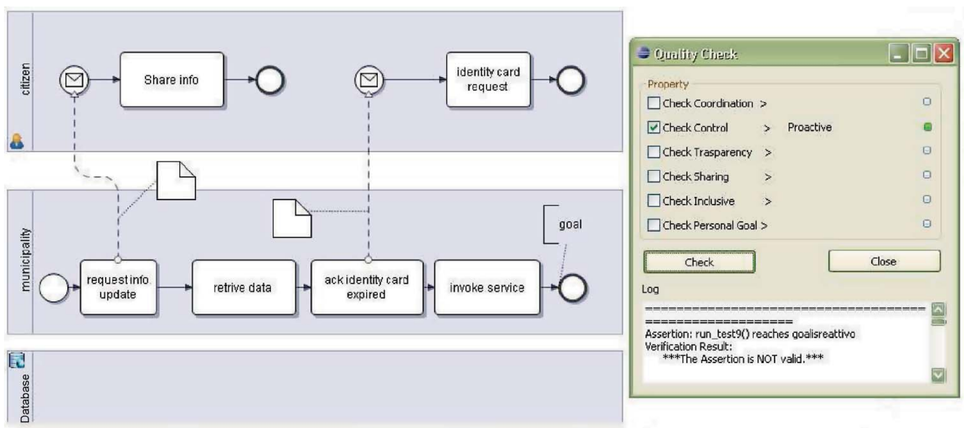


FIG. 6. Proactive control level.

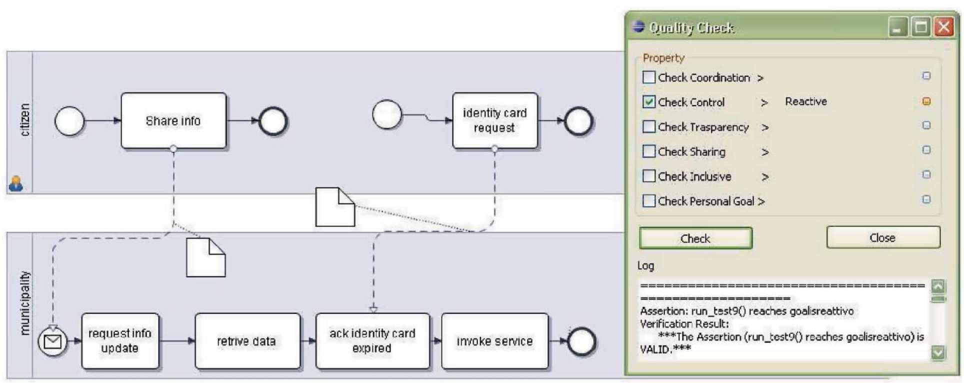


FIG. 7. Reactive control level.

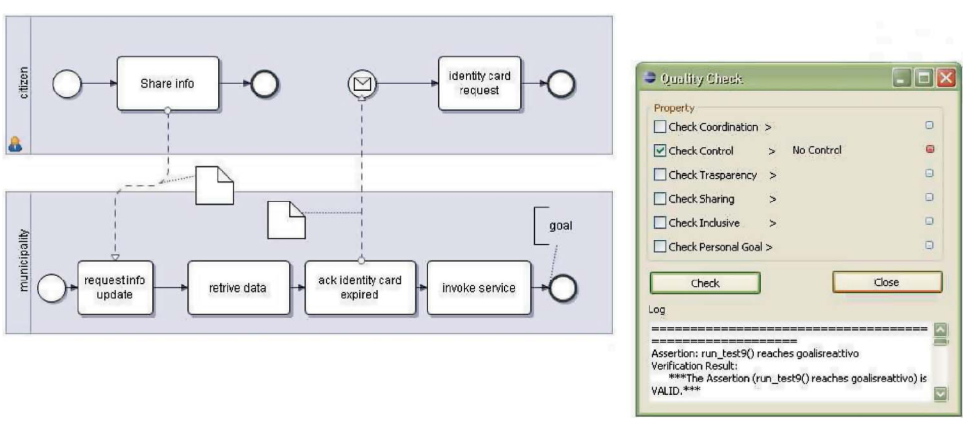


FIG. 8. No specific control level.

check that the type of participants included in the delivery matches this specific role, and we request interactions with it via messages exchange. Figure 9 shows how sharing is checked on a simple Business Process via BP4PA user interface. In this case, the requested pool is included and interactions are observable.

The *transparency quality* requirement predicates over the visibility of the Business Process put in place by the Public Administrations involved in the service delivery. By transparency, we mean the ability of the administration to make citizens aware of the delivery process and of its execution state, improving the citizens' perceived trust in this way. To check coordination levels, we assume that activities involved in the process delivery are typed with a transparency characteristic when they are observable. Lane can be also included in the process specification when roles of those in charge of completing such activities are clearly defined. These two requirements fit with the the levels of transparency recognized in the analysis phase, which are activity aware and role aware, respectively. Our verification on transparency concludes that the service reaches the quality goal linked to the activity aware transparency level only if all tasks involved in the process delivery in charge of the involved administration are traced and visible to the citizen. At the same time, if lanes are included in pools that are not citizens, mediators or a knowledge repository, then the role aware specification is observable after checking. In our modeling, we further considered a way of implementing and recognizing activity-aware transparency. In this case, we assume the availability of a particular pool called mediator specifically devoted to trace the state of the executed process. Our verification on transparency concludes that a service reaches the quality goal linked to the activity aware transparency if the mediator pool is included in the BP, and if tasks give specific feedback to the mediator via message exchanges. Nothing may be assumed about the role aware transparency in the case where the mediator is included. To provide simple examples on the transparency level, we refer to the case in Figure 10, where role transparency is implemented.

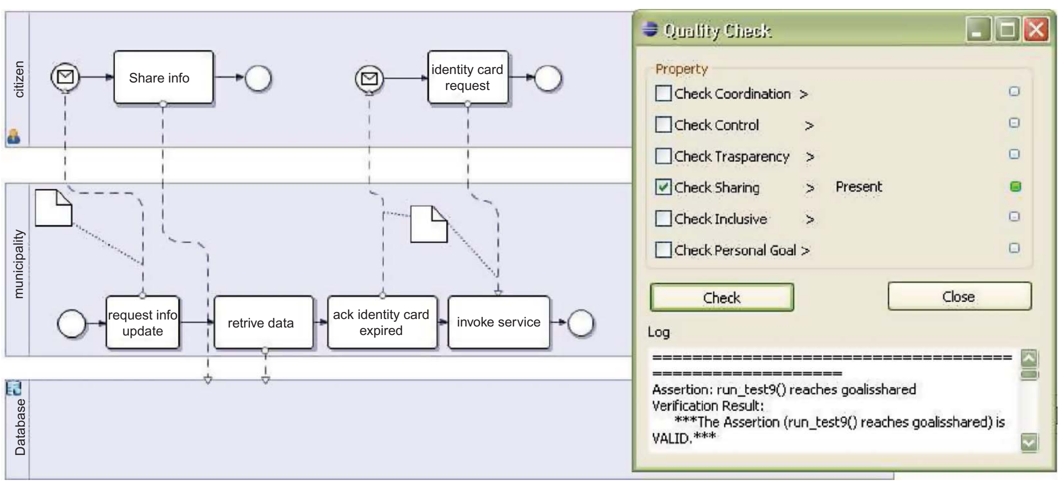


FIG. 9. Sharing check: data sharing.

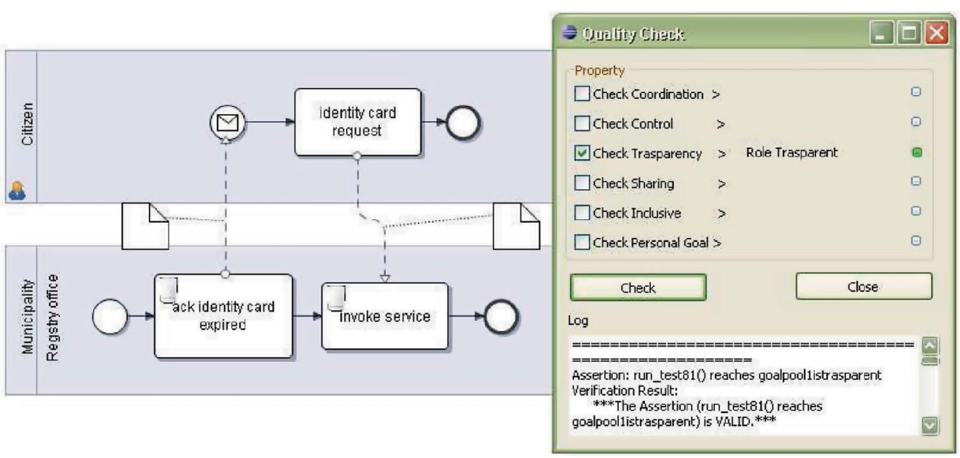


FIG. 10. Transparency check: role transparency.

The *inclusion quality* requirement predicates over the process of the involved Public Administrations in the delivery. In particular, with inclusion we mean the ability of the administration to provide service to citizens also considering their possible disability. To check inclusion levels, we assume that the gateway is typed with a level of inclusion that is observable in the process. Such gateway concerning possible interaction channels, user profiles, and internationalization can be identified during the analysis phase by domain experts. Our verification on inclusion concludes that the service reaches the quality goal linked to the specific inclusion level if in the process is allowed to switch between profiles, languages, and channels, respectively. To provide simple examples on the inclusion level, we refer to the cases in Figure 11, where all the three inclusion levels are introduced.

5. A CASE STUDY: THE NEWBORN REGISTRATION SERVICE

This section illustrates how the proposed approach can be used in practice on a real case study. This is the result of a close cooperation between our research group and a local Public Administration, where the proposed approach is currently applied in practice and tested on real processes. As for any approach using model checking techniques, it is worth mentioning that so far we experimented with 35 different processes, and all of them have generated relatively small state sets. In particular, the experiments we have conducted using a desktop PC equipped with a Core 2 Duo 2,20GHz and 4GB RAM, have highlighted that a process can be checked with respect to the properties included in the framework in less than three hours for the most complex BP scenarios. Moreover, the most complex BP we have analyzed so far generated a state space of around three millions states. This data seems to support the idea that in the current status (i.e., complexity of BP processes in the e-government domain, mapping we have defined and quality properties to be checked), the approach is applicable in real scenarios and can be a useful support for the BP designer.

The GDS under analysis refers to the newborn registration service, which is part of the wide area of cooperation among civil registration services (which are managed locally by municipalities). The service intends to permit newborn citizen registration, to get certificates, and any other services, regardless of their geographical location. In particular, the service under analysis supports in the most confortable way the registation of a baby's birth delivery, the request for the birth certificate, and the request for the fiscal code number at the same time. Such services also support the alignment of the information in all the public administration offices dedicated to trace information on newborn babies. The participants involved in such process are reported below and shown in Figure 12.

- The municipality where the baby has to be registered.
- The Ministry of Home Affairs, in Italian "Ministero degli Interni" is deputed to collect and to maintain up-to-date the information related to citizens. To carry out this task, it implements two different infrastructures:
  - SAIA is the technological infrastructure used by all the Italian Public Administrations to support information exchange concerning citizens data;
  - INA is a national knowledge base system, which contains information related to Italian citizens. In particular, it contains citizen's family name, first name, fiscal code, gender, place of birth, date of birth, and code of the municipality where the citizen lives.
- Tax office, in Italian "Agenzia delle Entrate," is the national organization in charge of issuing the Italian tax code card, officially known in Italy as *Codice Fiscale* (similar to a Social Security Number card

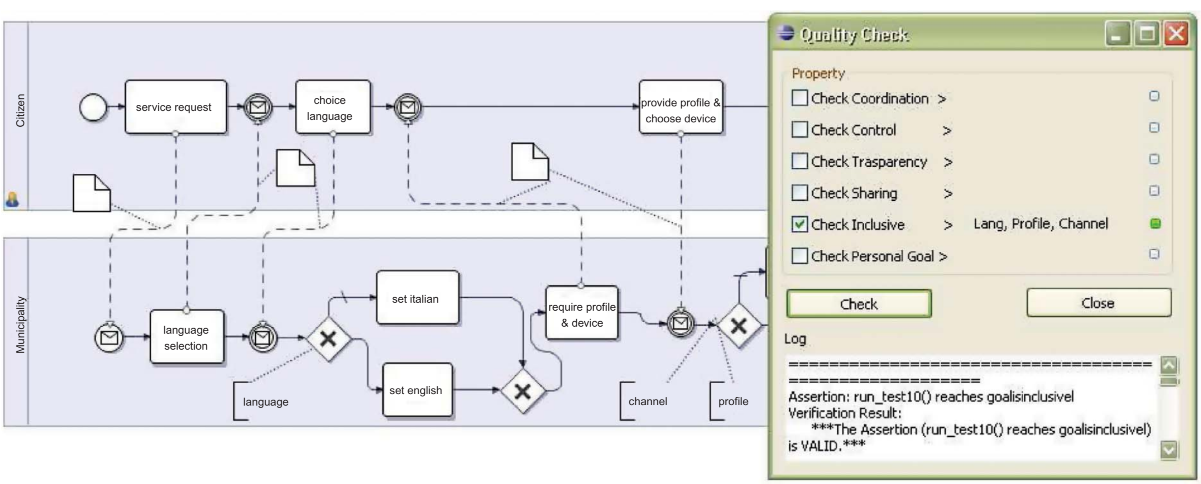


FIG. 11. Inclusion check: channel inclusiveness.

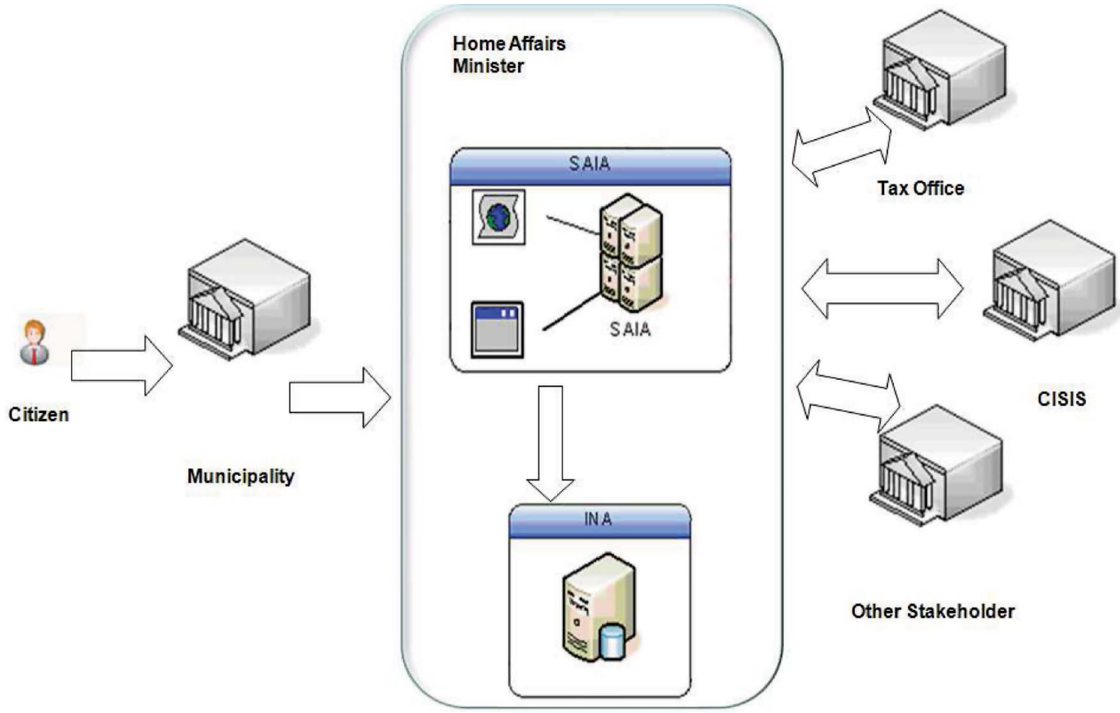


FIG. 12. Architecture of the newborn registration service.

in the United States). The card serves to identify—individuals residing in Italy unambiguously for tax related purposes.

- CISIS (Italian inter-regional center for information, statistical and geographical systems) is the association of regional authorities, which among a list of several activities, has to collect all the information requested for statistical purposes.

Involving such stakeholders, the service is implemented as a six-step process as reported below.

- As the first step, the parents ask to activate and access the newborn registration service. It is worth mentioning that the access can be provided at the municipality office or via the Web when suitable authorization and authentication mechanisms are available.
- The municipality collects the birth registration information from the parents.
- The municipality sends such data to the Ministry via the SAIA infrastructure.
- The Minister communicates the necessary data to the tax office that generates and returns a new tax code.
- The Minister stores all the received information, concerning the citizen, within the INA repository.
- The Minister communicates the data, relevant for statistical purpose, to the CISIS.

In order to use the proposed approach, a domain expert will have to codify a BP for the GDS he/she wants to put in place and to verify. In our case, we considered the national specification for the newborn registration service as it was codified in BPMN by the local PA as shown in Figure 13. Successively, the BP4PA tool can be run, and it will return the different quality levels the current implementation of the BP fulfills. In particular, for the newborn registration service, BP4PA provided the following results:

- Concerning the coordination dimension the communication level was reported. The BP does not foresee any semantic integration among the partners, and the back offices were able to interact requiring some level of human intervention. This is consequence of the usage of the e-mail system as communication channel among civil servants.
- Concerning the control dimension, a reactive level was reported. No integration with other services concerning new births are implemented.
- Concerning the sharing dimension, the data sharing level was reported. This level is reached thanks to the facilities provided by the INA-SAIA infrastructure.
- Concerning the transparency dimension, no mechanisms were implemented. The citizens are not in any way notified about the proceeding of the process, and they cannot observe the execution of any step in the process.
- Concerning the inclusiveness dimension, no conditions were checked regarding specific capabilities or disabilities of the citizens.

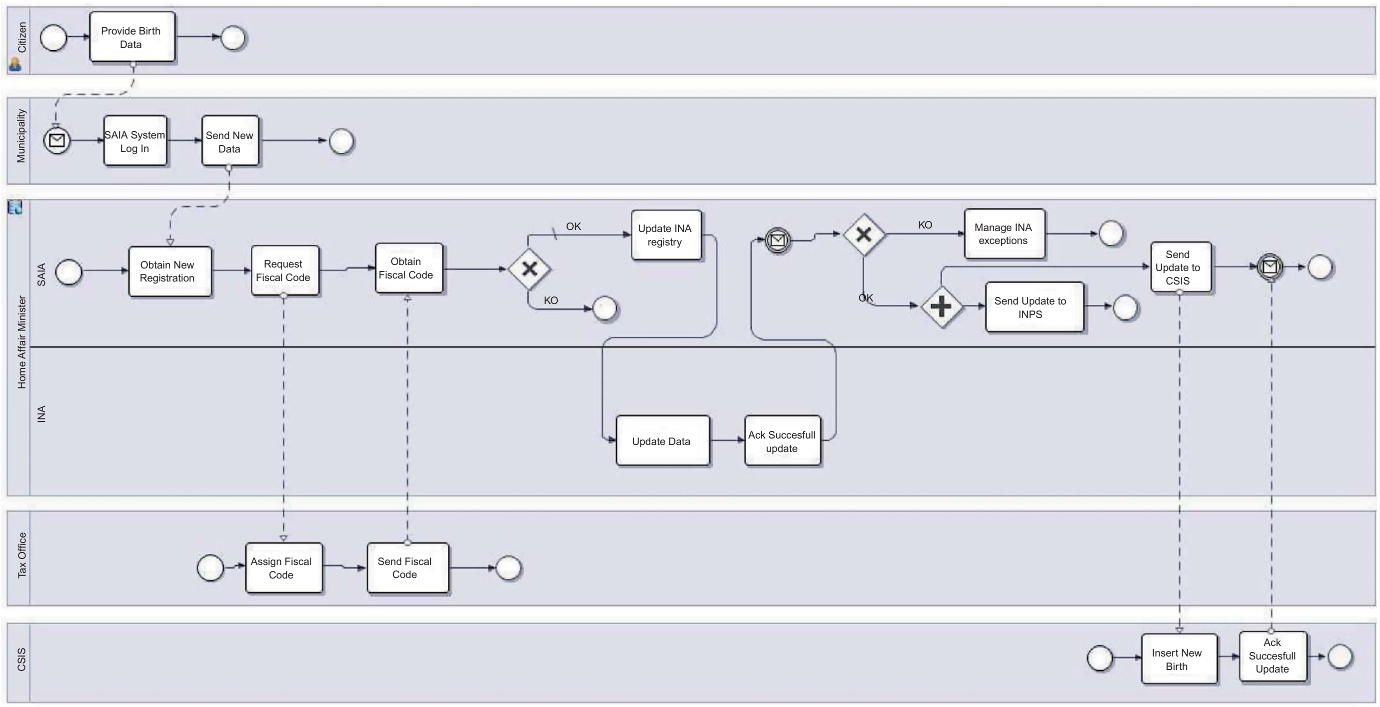


FIG. 13. Case study: Newborn Registration Service.

The analysis conducted using BP4PA permitted us to systematically discover many issues hidden within the BP specification that may have been resulted in low service usage. At the same time, the issues highlighted by the tool have been considered by the design team and domain experts in order to revise and improve the service delivery process, trying to fulfill, when possible, higher quality level.

6. RELATED WORKS

In the literature, there is a quite wide interest in applying techniques typically derived and used in the area of software development and evaluation, to the study of e-government organizations and processes. A first interesting discussion on the topic can be found in (Davies, Janowski, Ojo, & Shukla, 2007). This article is the result of a tutorial-workshop event on technological foundations of electronic governance. The event explored relevance and opportunities for the application of mature formal techniques based on mathematical theories and supported by industry-ready tools and methods to build technical solutions for e-government.

Other proposals provide user-friendly techniques hiding the complexity of formal languages and tools. There are, for instance, interesting approaches aiming at making model checking accessible to a large audience—even for people that are not trained in formal techniques. A recent survey on Business Process verification provides an interesting classification of proposed techniques (Morimoto, 2008). Further work in this area that is not included in the survey are classified according to the formal model they use: automata (Fu, Bultan, & Su, 2004), Petri Nets (Wohed, van der Aalst, Dumas, & ter Hofstede, 2003; Dijikman, Dumas, & Ouyang, 2007; Narayanan & McIlraith, 2002; Hamadi & Benatallah, 2003; Ye, Sun, Song, & Wen, 2008; Wynn, 2009), and process algebras (Salaün, Bordeaux, & Schaerf, 2004; Farrell, Sergot, & Bartolini, 2007; Stefansen, 2005; Zhao, Li, Liu, & Du, 2009; Wong & Gibbons, 2007; Janssen, Mateescu, Mauw, Fennema, & Stappen, 1999; Forster, Engels, Schattkowsky, & van der Straeten, 2007).

The approach that seems to be more related to our work, at least for what concerns the definition of a mapping from BPMN to a formal notation, is discussed in (Wong & Gibbons, 2007). In such an article, the authors demonstrate how a process algebra based on CSP notation can be applied to model complex workflow systems. The authors have given a formal semantics for BPMN in CSP using Z notation (Wong & Gibbons, 2008a), and use it to formally check compatibility of BPMN processes (Wong & Gibbons, 2008b). Nevertheless, the approach does not introduce a user-friendly and integrated environment for verification purposes. On the technical side, the approach uses the FDR model checker, and the ILOG BPMN modeler as stand alone tools. On the methodological side instead, the approach suffers of some mapping problems. Moreover, it can generate unbounded state spaces when BPMN loop activities are considered, and it does not include a formal semantic specification for the BPMN messages flows. Finally, the objective of such a work is not the definition of any domain related quality framework for BP evaluation, and the verification step just considers typical properties such as deadlock and liveness.

To the best of our knowledge, our approach is the first attempt, within the e-government domain, that tries to provide an easy to use environment both for BP design and BP evaluation with respect to a precisely defined quality framework.

7. CONCLUSIONS AND FUTURE WORK

This work has been conceived after having noticed the huge gap existing between the availability of GDSs and their real usage. We have then considered the GDS delivery phase, and we thought that some interesting issues were hiding in such a place. Our study led us to the definition of a framework defining a multidimensional space used to resume and classify available GDS in consideration of the processes used for their delivery. In this article, we have shown how it is possible to use formal techniques to conduct the analysis and then to assess the delivery processes of GDSs. In particular, the approach we propose permits an automatic check if a designed delivery process satisfies defined quality requirements, or instead if it suffers of some degrading characteristics.

The general idea has been also concretely implemented in a tool-set for the Eclipse platform. This permits to have an integrated development environment in which to model and evaluate a BP under development. The approach is currently used and under evaluation on real case studies in collaboration with our regional PA.

The current implementation of the approach has been conceived to make the extension of the framework with the definition of new quality properties easy. The extension of the framework is one of the research line in which we have started to try to identify additional interesting properties related to the delivery of GDSs. We also intend to extend the study to Business Processes related to other GDS aspects and PA activities. In particular, we started to investigate how to automatically assess if a BP can be put in place given constraints related to the resources available within a public office.

Finally, our work has been based on BPMN version 1.1, and we are working to adapt it taking into account the new features of BPMN 2.0.

ACKNOWLEDGMENTS

The authors would like to thank Damiano Falcioni for his support in the development and setting of the BP4PA tool-set. The authors also thank Regione Marche local Public Administration for partially funding this research work.

AUTHOR BIOS

Flavio Corradini is a full Professor of Computer Science at the University of Camerino. He received a master degree in computer science from the University of Pisa, and a PhD in computer engineering from the University of Rome “La Sapienza.” His research interests are centered around design methods, formal and semi-formal specification, and verification of complex systems, e-government, and information society. He is the coordinator of “UEG—Unicam e-government research group” and “COSY—Complex systems” research group. He collaborates with several companies and public administrations for the development of innovation technology, technology transfer, and scientific research.

Andrea Polini is an Assistant Professor at University of Camerino, and a Research Associate at ISTI-CNR in Pisa. Previously he has been Post-Doc at ISTI-CNR. He is a member of the “UEG—Unicam e-government research group,” and his research is mainly in the area of Verification and Validation of Software Systems—in particular in the context of Component Based Software and Service Oriented Architecture.

Alberto Polzonetti is an Assistant Professor of Computer Network and Service Oriented Computing at the University of Camerino. He is a member of the “UEG—Unicam e-government research group,” and he is active in several e-government projects related to services infrastructure and broadband development and diffusion. His research activity is focused on the study and on the introduction of technological advanced, innovative, and interoperable solutions able to encourage the Information and Communication Technology inside the Information Society.

Barbara Re is a post-doctorate at the Computer Science division, School of Science and Technology at the University of Camerino. She received a cum laude bachelor’s degree, a masters degree in computer science, and a PhD in Computer Science and Complex Systems from the University of Camerino. She is member of “UEG—Unicam e-government research group,” and her research interests include quality in e-government service, models, and languages for e-government interoperability and methodologies and technologies for e-government development. She is involved in collaborative research efforts with several public and private organizations.

REFERENCES

Baeten, J. C. M. (2005). A brief history of process algebra. *Theoretical Computer Science*, 335 (2–3):131–146.

Bavec, C. (2008). On stimulus for citizens’ use of e-government services. In *International Multiconference on Computer Science and Information Technology* (pp. 391–395). Wisia.

Bergstra, J., & Klop, J. (1982). Fixed point semantics in process algebra. Technical Report IW 208, Amsterdam, Mathematical Centre.

Berntzen, L., & Olsen, M. G. (2009). Benchmarking e-government — a comparative review of three international benchmarking studies. In *ICDS* pp. 77–82. Cancun, Mexico: IEEE Computer Society.

Carnegie Mellon University—School of Computer Science (1998). SMV. Retrieved from http://www.cs.cmu.edu/~modelcheck/

Clarke, E. M., Grumberg, O., & Peled, D. A. (2000). *Model Checking*.

Colclough, G. (2007). The user challenge benchmarking the supply of online public services—7th measurement. Technical report, prepared by: Capgemini. European Commission Directorate General for Information Society and Media. Bruxelles.

Colclough, G., & Tinholt, D. (2009). Smarter, faster, better e-government - 8th benchmarking measurement. Technical report, prepared by: Capgemini, RAND Europe, IDC, SOGETI and DTi. European Commission Directorate General for Information Society and Media. Bruxelles.

Corradini, F., Falcioni, D., Polini, A., Polzonetti, A., & Re, B. (2009a). From bpmn to csp — toward business process verification for e-government service delivery. Technical report, University of Camerino, Camerino, Italy.

Corradini, F., Hinkelmann, K., Polini, A., Polzonetti, A., and Re, B. (2009b). C2ST: a quality framework to evaluate e-government service delivery. In M. Wimmer, H. J. School, M. Janssen, & R. Traunmüller (Eds.), *8th International Conference EGOV 2009, Proceedings of ongoing research, project contributions and workshops* (pp. 74–84). Linz, Austria: Trauner.

Davies, J., Janowski, T., Ojo, A. K., & Shukla, A. (2007). Technological foundations of electronic governance. In T. Janowski, & T. A., Pardo, (Eds.), *ICEGOV*, volume 232 of *ACM International Conference Proceeding Series*, (pp. 5–11). ACM, Macao.

Dhillon, G., & Mishra, S. (2007). Information systems security governance research: A behavioral perspective. In *2nd Annual Symposium on Information Assurance*.

Dijkman, R. M., Dumas, M., & Ouyang, C. (2007). Formal semantics and analysis of BPMN process models using petri nets. Retrieved from

Emerson, E. (1990). *Temporal and modal logic, Handbook of theoretical computer science (vol. B): formal models and semantics* (p. 9951072). Cambridge, MA.

Farrell, A. D., Sergot, M. J., & Bartolini, C. (2007). Formalising workflow: a CCS-inspired characterization of the YAWL workflow patterns. *Group Decision and Negotation*, 16 (3): 213–254.

Forster, A., Engels, G., Schattkowsky, T., & van der Straeten, R. (2007). Verification of business process quality constraints based on visual process patterns. In *TASE '07: Proceedings of the First Joint IEEE/IFIP Symposium on Theoretical Aspects of Software Engineering*, pages 197–208, Washington, DC, IEEE Computer Society.

Fu, X., Bultan, T., & Su, J. (2004). Analysis of interacting BPEL web services. In *WWW '04: Proceedings of the 13th international conference on World Wide Web* (pp. 621–630). New York, NY, ACM.

Hamadi, R., & Benatallah, B. (2003). A Petri net-based model for web service composition. In *ADC '03: Proceedings of the 14th Australasian database conference* (pp. 191–200). Darlinghurst, Australia, Australian Computer Society, Inc.

Harmon, P. (2004). *Business Process Change—A guide tor Business Manager and BPM and Six Sigma Professionals*. USA: Horgan Kaufmann.

Hoare, C. A. R. (2004). *Communicating Sequential Processes*. Prentis.

Holzmann, G. J. (1997). The model checker SPIN. *IEEE Transactions on Software Engineering*, 23 (5): 279–295.

Janssen, W., Mateescu, R., Mauw, S., Fennema, P., & Stappen, P. V. D. (1999). Model checking for managers. In D. Dams, R. Gerth, S. Leue & M. Massink (Eds.), *Proceedings of the 5th and 6th International SPIN Workshops on Theoretical and Practical Aspects of SPIN Model Checking* (pp. 92–107) London, UK, Springer-Verlag.

Lindsay, A., Downs, D., & Lunn, K. (2003). Business processes-attempts to find a definition. *Information and Software Technology*, 45 (15): 1015–1019.

Mendling, J. (2008). *Metrics for Process Models*. Germany: Springer.

Morimoto, S. (2008). A survey of formal verification for business process modeling. In *ICCS '08: Proceedings of the 8th international conference on Computational Science, Part II* (pp. 514–522) Berlin, Heidelberg, Springer.

Narayanan, S., & McIlraith, S. A. (2002). Simulation, verification and automated composition of web services. In *WWW '02: Proceedings of the 11th international conference on World Wide Web* (pp. 77–88). New York, NY, ACM.

Norris, P. (2001). *Digital Divide Civic Engagement, Information Poverty, and the Internet Worldwide*. Cambridge University Press.

Salaün, G., Bordeaux, L., & Schaerf, M. (2004). Describing and reasoning on web services using process algebra. In *ICWS '04: Proceedings of the IEEE International Conference on Web Services* (p. 43). Washington, DC, IEEE Computer Society.

Salem, F. (2007). Benchmarking the e-government bulldozer: beyond measuring the tread marks. *Measuring Business Excellence*, 11 (4): 9–22.

Stefansen, C. (2005). SMAWL: a SMAII Workflow Language based on CCS. In *CAiSE Short Paper Proceedings*.

Sun, J., Liu, Y., & Dong, J. S. (2008). Model checking CSP revisited: introducing a Process Analysis Toolkit. In T. Margaria, & B. Steffen, (Eds.), *ISoLA*, volume 17 of *Communications in Computer and Information Science*, (pp. 307–322). Berlin Heidelberg: Springer.

van der Aalst, W. M. P., & ter Hofstede, A. H. M. (2005). YAWL: Yet Another Workflow Language. *Inf. Syst*., 30 (4): 245–275.

Wauters, P., & Colclough, G. (2006). On-line availability of public services: How is europe progressing? Web based survey on electronic public services report of the 6th measurement June 2006. Technical report, Prepared by Capgemini. European Commission Directorate General for Information Society and Media. Bruxelles.

White, S. A., & Miers, D. (2008). *BPMN Modeling and Reference Guide Understanding and Using BPMN*. Future Strategies Inc.

Wohed, P., van der Aalst, W.M.P., Dumas, M., & ter Hofstede, A.H.M. (2003). Analysis of web services composition languages: the case of BPEL4WS. In I.-Y. Song, S. W., Liddle, T. W., Ling & Scheuermann, P. (Eds.), *ER*, volume 2813 of *Lecture Notes in Computer Science* (pp. 200–215). Berlin Heidelberg: Springer.

Wong, P. Y. H., & Gibbons, J. (2007). A process-algebraic approach to workflow specification and refinement. In M. Lumpe & W. Vanderperren, *Proceedings of 6th International Symposium on Software Composition*, SC 2007, Braga, Portugal, March 24–25, 2007, Revised Selected Papers, volume 4829 of *LNCS*. (pp. 51–65) Berlin Heidelberg: Springer Verlag.

Wong, P. Y., & Gibbons, J. (2008a). A Process Semantics for BPMN. In S. Liu, T. Maibaum, & K. Araki (Eds.), *Proceedings of 10th International Conference on Formal Engineering Methods*., volume 5256 of *LNCS*. (pp. 355–374) Berlin Heidelberg: Springer Verlag. Extended version available at http://web.comlab.ox.ac.uk/oucl/work/peter.wong/pub/bpmnsem.pdf

Wong, P. Y. H., & Gibbons, J. (2008b). Verifying business process compatibility (short paper). In *QSIC '08: Proceedings of the 2008 The Eighth International Conference on Quality Software* (pp. 126–131). Washington, DC, IEEE Computer Society.

Wynn, M., Verbeek, H., van der Aalst, W., ter Hofstede, A., & Edmond, D. (2009). Business process verification finally a reality! *Business Process Management Journal*, 15 (1): 74–92.

Ye, J., Sun, S., Song, W., & Wen, L. (2008). Formal semantics of BPMN process models using YAWL. In *IITA '08: Proceedings of the 2008 Second International Symposium on Intelligent Information Technology Application.* (pp. 70–74) Washington DC: IEEE Computer Society.

Zhao, L., Li, Q., Liu, X., & Du, N. (2009). A modeling method based on CCS for workflow. In *ICUIMC '09: Proceedings of the 3rd International Conference on Ubiquitous Information Management and Communication.* (pp. 376–384) New York: ACM.

zur Muehlen, M., & Recker, J. (2008). How much language is enough? Theoretical and practical use of the business process modeling notation. In Z., Bellahsene, & M. Léonard, (Eds.), *CAiSE*, volume 5074 of *Lecture Notes in Computer Science* (pp. 465–479). Berlin Heidelberg: Springer.