Embedding 'Break the Glass' into Business Process Models

Silvia von Stackelberg, Klemens Böhm, and Matthias Bracht

Karlsruhe Institute of Technology (KIT), 76131 Karlsruhe, Germany

Abstract. Break the Glass (BTG) is an important feature for authorization infrastructures, as it provides flexible access control in exceptional cases. Current realizations have two drawbacks: They neglect the need to manage authorization steps, and they do not take immediate process context into account. Our approach in turn embeds BTG functionality into business processes (BPs): The steps to perform BTG and the obligations compensating a BTG access for data are parts of the BPs. To support process designers in embedding BTG steps and obligations, we introduce an expressive annotation language for specifying BTG tasks for BP models. In particular, our language allows process designers to take BP context into account and to specify security constraints for role holders performing BTG tasks. Using our approach, one can efficiently specify and use context-aware BTG functionality for BPs.

Keywords: Security, process model annotation language, immediate context.

1 Introduction

Problem Statement. Security mechanisms are important for Business Process Management (BPM). For instance, authorization constraints specify which roles may perform a task or access certain data. However, such mechanisms sometimes are too rigid, and more flexibility is needed. To illustrate, emergencies (e.g., in E-health) and disaster management necessitate rights to access data in exceptional situations. Thus, a trade-off between security on the one hand and flexibility on the other hand needs to be facilitated.

The so-called *Break the Glass (BTG)* principle provides flexibility by allowing users to overcome access denials in exceptional cases [1]. The designer specifies in advance who, in particular situations, will have access rights he normally does not have. In line with [5], the prerequisites to "break the glass" from the application perspective are: (1) regular access is denied, (2) BTG access is foreseen for the exceptional case, (3) a user explicitly asks for access, (4) optionally, another user has to agree to this access. We call the sequence of steps when users ask for exceptional access *BTG steps* in the following. Next, *obligations* typically are part of BTG, i.e., operations that compensate[1] for the security violations. Obligations can be triggered immediately after breaking the glass (synchronously) or later (asynchronously).

[1] We use the term *compensation* for the execution of obligations. As a data access cannot be undone, it is at least mitigated by compensating actions.

R. Meersman et al. (Eds.): OTM 2012, Part I, LNCS 7565, pp. 455–464, 2012.

*Example 1 (***E-Health).** In the regular case, only dedicated persons, such as the family doctor of a patient, are authorized to access health-record data of patients. We assume that this data is stored externally and policies (e.g., sticky policies) specify authorizations for data access. In a life-threatening situation, other members of the medical staff might need access to the data. By *Breaking the Glass*, physicians who are not authorized in the regular case access the record in a controlled way. An exceptional access results in many obligations, such as auditing the data access, informing the familiy doctor, among others. We exemplarily focus on **O1:** At the end of the treatment process, the physician has to send a report to the family doctor. Here, the point of time when sending the report depends on the BP context, i.e., when the treatment of the patient has been finished. This obligation is asynchronous because it refers to a later point in time.

We envision integrating BTG functionality into BPMS. This is new and challenging, because existing approaches providing authorization infrastructures for BTG (e.g., [1], [5], and [9]) do not cover the following aspects: (1) Modelling BTG steps and obligations as part of the BP and executing them. (2) Considering BP-specific features, BP context in particular.

Regarding (1), related work leaves the execution of BTG steps and obligations to the application and views them as black boxes. However, a BTG access typically consists of several steps. The same holds for obligations. This asks for mechanisms to embed BTG steps and obligations into the BP, since the modelling of such steps and their execution is exactly the purpose of BPMSs.

The development of context-aware systems is a challenging research area. Most approaches take environmental context into account. *Immediate process context* in turn is information that characterizes the process itself. It refers to the execution state of a process instance, such as the state of tasks, associated actors, or objects to be accessed [12]. Regarding (2), combining immediate BP context with BTG functionality has several advantages, as we will explain in Section 2. But existing work does not take immediate BP context for BTG realizations into account.

Goals and Challenges: Our overall goal is to integrate BTG functionality into the BP and to have it executed by a BPMS. By doing so, we take BP context into account. To accomplish this, this paper focuses on the following goals:

- *Facilitating the embedding of BTG steps and obligations into BPs.* Without any support for the embedding, process designers have to model both the process logic and the security constraints for BTG functionality by hand. This requires profound security knowledge and thus is error prone; and it is time-consuming. Thus, there should be support at the process-modelling level. By using annotations for process models (e.g., [7], [11], and [14]), designers can rely on the modelling primitives they are used to. We develop an annotation language for BP models representing BTG functionality.
- *Context-aware annotation language.* As BP context is important for BTG functionality, the annotation language has to provide support for the coupling of contextual information with BTG tasks.

We leave the design and realization of an infrastructure supporting context-aware BTG to a future publication.

The goals lined out above are challenging, for the following reasons:

- As the embedding of BTG functionality into BP is new, the design of an expressive annotation language asks for a systematic requirements analysis.
- This results in the specification of a comprehensive set of expressions to represent BTG functionality (e.g., which tasks have to be performed, who is authorized).
- Current systems do not feature the coupling of immediate BP context with BTG functionality. To support this, we develop a representation of BP context process dessigners can easily use.

Contributions: We have developed new concepts to embed BTG functionality at the BP modelling layer. "Embedding" means that potential BTG steps and obligations are integrated into a BP, and BP context is taken into account. In particular, we make the following contributions:

- *Motivation.* We list advantages of using immediate context information for BTG functionality. As context can be any information, we provide expressions for the specification of context relevant for an application.
- *Specification of an annotation language allowing to represent BTG steps and obligations.* In particular, it allows to specify actors involved in BTG steps and BP-context-specific constraints for BTG options. By using these constructs, process designers can smoothly embed BTG support into BPMS. We specify a generic process fragment the BPMS has to execute in order to fulfill the specifications contained in annotations.

Paper structure: We motivate context-aware BTG functionality in Section 2. Section 3 lists requirements. Section 4 describes the annotation terms for BP models. Section 5 discusses related work, and Section 6 concludes.

2 Motivation for Contextual BTG Functionality

Immediate BP context relevant for BTG can be information on the core BP regarding the functional, behavioral, organizational, operational, and data aspects of process instances. We borrow these categories from [12]. – Coupling BTG functionality with BP context has the following advantages:

(1) BTG steps may comprise BP-context-specific constraints: BP schemas allow to impose control-flow constraints on BTG steps. For example, in the E-health scenario, there might be a task to determine the urgency of a treatment. A physician might be allowed for BTG only if this task has been performed. Such a constraint can be expressed by referring to the *execution state* of a BP instance. However, existing BTG approaches do not allow to specify this.

(2) Specification of constraints for obligations: Constraints for the execution of obligations can be specified in the same way as for BTG steps. In our scenario,

there might be one or several physicians involved in an treatment. An example of a constraint is to send an email only when several physicians have been involved.

(3) Specification of BP context for obligation parameters: In general, obligations are parameterized. For example, an obligation might say that an individual who accesses a data object in parallel to a BTG access on this data must be informed about the respective BTG action. In our example, the system must pass the email address of that individual to the application executing the obligation. By using BP-context information on *associated actors, the system* might determine the receivers of the email automatically.

(4) Triggering asynchronous obligations: Synchronous obligations are triggered immediately, together with the BTG action. Asynchronous obligations are triggered at an absolute or a relative point in time. In our example, the *execution state of a BP* determines when an obligation takes place (i.e., send the patient report when the last task of the treatment is finished). Being able to refer to execution states of tasks gives way to asynchronous obligations.

As these advantages are essential, it is important to combine BTG functionality with BP context.

3 Requirements for Annotation Language

To identify requirements on a language allowing to specify BTG functionality, we have analyzed BTG use cases in two different real-world scenarios, an E-employment and an E-health application. Following these analyses, a BTG-annotation-language must support the following aspects:

R1: *Security constraints for BTG users:* BTG functionality has to be provided in a controlled way, i.e., it must be specified at design time who shall obtain the BTG rights, namely to break the glass, to access data, and to repair the glass. Thus, the BTG vocabulary must distinguish different types of users involved in a BTG action. The vocabulary must allow to specify authorization and authentication constraints for these users.

R2: *BP-context-specific constraints:* It must be possible to specify the start time for BTG steps or obligations, i.e., *when* the tasks have to be performed, by taking the BP context into account. In particular, asynchronous obligations that rely on BP-context-specific conditions must be possible. Example 1 has motivated this. Further, it should be possible to represent conditions for the execution of BTG steps and obligations, i.e., *whether* tasks have to be performed. To illustrate, one might specify that an obligation is needed only if an external physician has worked on the emergency treatment. The specification of BP context must be user-friendly [6]. This means that process designers should not have to deal with the BP-engine-internal representation of BP context, but should be able to specify BP context at the abstraction level of BP models.

R3: *Parameters for obligations:* It must be possible to specify BP-context-specific parameters of obligations (e.g., associated actors), cf. Example 1.

4 Design

In this section we describe how we embed options for breaking the glass into BP models. We first describe the different BTG roles and motivate annotating process models with BTG functionality. We then say how we represent BP context. Finally, we describe the annotation language in brief.

4.1 BTG Roles

We introduce roles having BTG rights in the following. In line with [2], we distinguish three types of users involved in a BTG option: the first type are users who have the right to break the glass, i.e., users who activate a BTG case (e.g., patients who decide). We call the corresponding role *BTG Activator Role*. Second, the *BTG Access Role* are users who have the right to access a resource if BTG has happened (physicians in our example). In practice, it is possible that process participants have both rights. Third, the *BTG Compensator Role* are users who are allowed to perform obligations in the BTG case.

4.2 Embedding BTG Functionality into Business Processes

Our idea is to integrate BTG steps into the application process. Thus, the BP Engine controls their execution.

The question now is how to realize this embedding. We see several alternatives, with two extremes: to represent them within the process model or to dynamically adapt process instances at runtime if needed (ad-hoc adaptation).

With the first extreme, process designers embed any options for breaking the glass in the BP model, using conventional modelling primitives. Process events and gateways can represent these options for exception handling. This means that a process instance can perform any BTG case or not. This is likely to lead to very complex BP models, because a single BTG case already consists of a sequence of tasks and might have many corresponding obligations. However, BTG functionality is only needed in exceptional cases, and whether it is needed is known only at runtime. This observation leads to the second extreme, namely to enable ad-hoc changes at runtime, meaning that process instances deviate from the specified process model. This can be interpreted as a case of exception handling, and it affects only single process instances. Process designers have to specify allowed deviations in advance. This approach requires a BP Engine that is capable to deal with ad-hoc changes at runtime. Currently, there is only little support in BP Engines for this (e.g., by the AristaFlow BPM Suite [3]). If platform-independence is an issue, a solution currently cannot rely on these features.

Design Decision: Our approach is a middle ground. We let process designers specify BTG options in a BP model[2] with specific annotation vocabulary. The

[2] In line with our security-annotation language, we represent BTG annotations in BPMN process models. But our BTG approach is sufficiently general to be applied to other process-model-languages.

BPMS transforms these annotations by extending the BP schema with canned process fragments and executes them as part of the BP. By means of annotation terms, process designers specify who will have the various BTG rights for data and the constraints for enabling BTG.

4.3 Formalizing BP Context

It is the task of the Engine to manage BP context. As BP context is important for BTG, we need a way to represent it in the BP model. One way is to specify constraints for BP context relevant for BTG by using the internal representation of the BP Engine. But this is error-prone and time-consuming, since process designers typically are not familiar with the internal representations.

Design Decision: We propose a vocabulary to represent BP context in BTG annotations. We formalize BP context on the abstraction level of BP models by introducing functions for tasks and data that return values representing the BP context. These functions enable the specification of associated actors, tasks, and data objects to be used for the representation of temporal and causal BP-context constraints in the annotations. The BPMS transforms these specifications into representations the BP Engine can handle. This addresses R2.

We define the syntax and semantics of BP contraints in [13]. In brief, a BP-context constraint is a Boolean expression, using at least one of the following functions:

- The functions `performer(task)`, `data-user(object)`, `owner(object)` return subjects related to a BP instance, namely the actor performing a task instance, the actor accessing a data object, and the data owner respectively.
- The functions `start-time-exec(task)` and `end-time-exec(task)` return the start and end times of the execution of a task.
- The functions `start-time-access(object)` and `end-time-access(object)` return the start time and end time of access to a data object.
- `data-access(task)` returns the set of data objects accessed by a task.

This set of functions is sufficient for the applications we have studied. Using these functions, process designers can represent a comprehensive set of BP-context-constraints within annotations for BTG steps and obligations.

*Example 2 (***BP Context Constraints).** To express that BTG is only allowed for adults, we set `exec=performer(activity-ID).age` $\geq$ `18`. We specify an asynchronous start time depending on the execution time of a task by `start` = `end-time-exec(activity-ID)`. To say that an obligation has to be executed if several performers are involved we formulate `exec` = `performer(activity-ID-1)` $\neq$ `performer(activity-ID-2)`.

4.4 Specification of BTG Steps for BP Models

Our annotation language features the specification of security aspects and of BP-context constraints as follows: To represent authorizations for BTG steps, we

need two out of three BTG roles, namely *BTG Activator Role* and *BTG Access Role*. This accounts for R1. These role holders have the right to perform particular BTG tasks. The specifications for `BTGActivator` and `BTGAccessor` can have optional authentication refinements. Further, one can specify constraints on the start or the execution of BTG steps by means of parameters `start` and `exec`. The first one states when the BTG steps have to be executed, and which constraint must hold at this point in time. In contrast, `exec` specifies constraints that must hold for executing BTG steps in general. One BTG action can have many obligations, and obligation specifications can be complex. Thus, annotations contain a list of obligation-IDs which have to be executed when the glass has been broken. We annotate each obligation separately for the BP model.

A BTG annotation, starting with "≪BTG:", contains a set of assignments for a specified vocabulary, and ends with "≫". The parameter `right` specifies the nature of the access to a data `object` for which the glass can be broken. The assignments for `object` and `right` are obligatory. [13] gives a complete definition of the syntax and the semantics.

*Example 3 (***E-health (cont.).** Figure 1 displays the excerpt of a process model representing the visit of a patient to a physician [13]. To enable BTG functionality for health records of patients, we make use of the BTG-annotation term for activity "Check health-record availability". Figure 1 graphs the annotation.

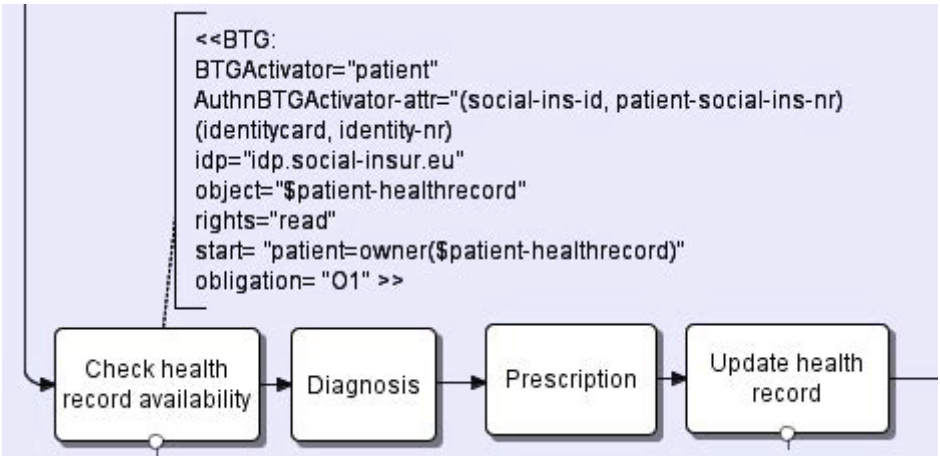

Fig. 1. Annotation of Activity "Check health-record availability"

The meaning of the annotation is as follows: To provide the BTG option, a process designer assigns patients whose health record might be accessed to *BTG Activator*. In our case, the patients have to agree to break the glass. As this is a security-relevant task, the process designer asks for authentication for the *BTG Activator*. In other words, the patient now has to authentify himself, and this needs to be modelled. The authentication specification says that the IdP must

authenticate a patient by the AuthnBTGActivator attributes social insurance number and identity-card number. The data to be accessed are health records of the patients. We set "read" access rights for the BTG case. The condition `start` specifies that the glass can only be broken if the patient is the owner of the data object. The parameter obligation specifies that O1 must be executed.

A BTG annotation means that the BPMS has to provide options for BTG steps, as described in Section 4.2. To embed these steps into the BP, we rely on the fundamental technique of process fragments, enabling to re-use parts of process structures. Using our approach, executable BPs require specifications for BP-context constraints as well as for security (data access, authorizations and authentications for role holders). The BTG annotations contain these specifications, and our secure BPMS transforms them to the extended process model. By doing so, process fragments are generic and can be used in any BP model.

Figure 2 shows the process fragment for BTG steps. It represents the execution order for BTG tasks as well as conditions on BP-context constraints. The annotated role assignments specify authorizations for role holders.

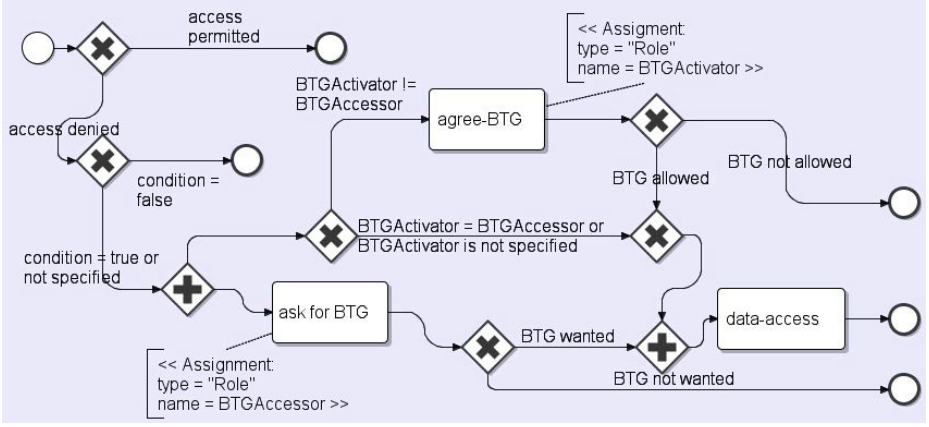

Fig. 2. Process Fragment for BTG steps

In line with annotations for BTG steps, we provide a vocabulary to describe obligation and represent them as language primitives. [13] gives a complete definition of the syntax and the complete semantics. The specification fulfills R3. Regarding the transformation, the system substitutes each BTG annotation with a BTG-process fragment, specifies the extended BP model, and generates the access-control policy.

5 Related Work

Regarding the integration of BTG into business processes, work from three research threads is of particular relevance: access-control policies for BTG, context-aware, security-related research for BPs, and security-annotation languages.

BTG access control policies: Several approaches realize BTG by implementing access control policies ([1], [8], [5] and [9]). But they do not feature support for BP context, due to their generic nature, and do not address the management of BTG steps and obligations from the perspective of the application, as we do. Our approach in turn does not focus on a BTG-enabled access-control-policy language, but on an infrastructure for embedding BTG functionality into business processes. We manage BTG authorization functionality to some degree by tasks being part of the BP. To illustrate, the BPMS executes process branches with BTG options, instead of offering BTG options by the authorization component, as in [5]. Our approach makes access control rules for BTG easier.

Context-aware security support: According to the classification in [12], our work focuses on immediate context. We thereby also consider security and privacy aspects. The activity and object context in [10] is similar to our understanding of immediate context. Most recent work on context-aware BP (e.g., [9]) is confined to the context of the *environment*, which we do not address. We employ BP context for security aspects. [10] summarizes well-known approaches on BP-context-aware access control methods. To bind access rights to the execution time of tasks (strict least privilege), the approaches use immediate BP context (e.g., [10]). Further, the realization of Binding and Separation of Duties [4] requires immediate context information at runtime. Our purpose differs from the discussed approaches. We address context-aware BTG functionality.

Security modelling languages: [7], [11], [14], among others, propose annotation languages to represent security constraints in BP models, but lack in two aspects: None of them takes BTG into account; None of them provides features to represent BP context as part of the annotation language. Our language is the first to cover this.

To our knowledge, our approach is unique in that we embed BP-context-constraints into the BTG-annotation-language, and use this contextual information for the embedding of process fragments by taking authorization rules into account.

6 Conclusions

The "Break the Glass" concept facilitates controlled access to data in exceptional situations. To our knowledge, this article has been first to provide BTG functionality for business processes. As breaking the glass and compensating a BTG action require several tasks, BTG steps and obligations should be embedded in processes. We have shown that using BP context for BTG tasks is essential.

To disburden the process designer from modelling BTG steps and obligations by hand, we have proposed a vocabulary for annotating the process model with BTG functionality. In particular, we take BP context into account. This reduces the design effort significantly.

Acknowledgements. This research has received funding from the Seventh Framework Programme of the European Union (FP7/2007-2013) under grant agreement n° 216287 (TAS^3 - Trusted Architecture for Securely Shared Services) as well as by the European Social Fund and by the Ministry Of Science, Research and the Arts Baden-Württemberg.

References

1. Brucker, A.D., Petritsch, H.: Extending Access Control Models with Break-glass. In: SACMAT (2009)
2. Chadwick, D. (ed.): Design of Identity Management, Authentication and Authorization Infrastructure, TAS3 Deliverable 7.1, Version 3.0.1 (2010)
3. Dadam, P., Reichert, M., Rinderle-Ma, S., Lanz, A., Pryss, R., Predeschly, M., Kolb, J., Ly, L.T., Jurisch, M., Kreher, U., Göser, K.: From ADEPT to AristaFlow BPM Suite: A Research Vision Has Become Reality. In: Rinderle-Ma, S., Sadiq, S., Leymann, F. (eds.) BPM 2009. LNBIP, vol. 43, pp. 529–531. Springer, Heidelberg (2010)
4. Bertino, E., Martino, L., Paci, F., Squicciarini, A.: Security for Web Services and Service-Oriented Architectures. Springer (2010)
5. Ferreira, A., Chadwick, D., Farinha, P., Correia, R., Zao, G., Chilro, R., Antunes, L.: How to securely break into RBAC: The BTG-RBAC model. In: ACSAC, pp. 23 –31 (2009)
6. Hallerbach, A., Bauer, T., Reichert, M.: Context-based configuration of process variants. In: TCoB, pp. 31–40 (2008)
7. Mülle, J., von Stackelberg, S., Böhm, K.: Modelling and Transforming Security Constraints in Privacy-Aware Business Processes. In: SOCA, pp. 1–4 (2011)
8. Alqatawna, J., Rissanen, E., Sadighi, B.: Overriding of Access Control in XACML. In: POLICY, pp. 87–95 (2007)
9. Marinovic, S., Craven, R., Ma, J., Dulay, N.: Rumpole: A Flexible Break-glass Access Control Model. In: SACMAT, pp. 73–82 (2011)
10. Park, S.H., Eom, J.H., Chung, T.M.: A Study on Access Control Model for Context-Aware Workflow. In: INC, IMS and IDC, pp. 1526–1531 (2009)
11. Rodríguez, A., Fernández-Medina, E., Piattini, M.: A BPMN extension for the modeling of security requirements in business processes. Trans. Inf. Syst. – IEICE E90-D, 745–752 (2007)
12. Rosemann, M., Recker, J.C., Flender, C.: Contextualisation of business processes. Int. Journ. of Business Process Integration and Management 3(1), 47–60 (2008)
13. von Stackelberg, S., Böhm, K., Bracht, M.: Embedding BTG into Business Processes (2012), `http://dbis.ipd.kit.edu/1860.php`
14. Wolter, C., Schaad, A.: Modeling of Task-Based Authorization Constraints in BPMN. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 64–79. Springer, Heidelberg (2007)