Interaction Protocols for Human-Driven Crisis Resolution Processes

Eric Andonoff, Chihab Hanachi(✉), Nguyen Le Tuan Thanh, and Christophe Sibertin-Blanc

University of Toulouse 1 Capitole, IRIT laboratory 2 rue du Doyen Gabriel Marty, 31042 Toulouse Cedex, France
{Eric.Andonoff,Chihab.Hanachi,Nguyen.Le, Christophe.Sibertin-Blanc}@irit.fr

Abstract. This work aims at providing a crisis cell with process-oriented tools to manage crisis resolutions. Indeed, the crisis cell members have to define the crisis resolution process, adapt it to face crisis evolutions, and guide its execution. Crisis resolution processes are interaction-intensive processes: they not only coordinate the performance of tasks to be undertaken on the impacted world, but they also support regulatory interactions between possibly geographically distributed crisis cell members. In order to deal with such an interweaving, this paper proposes to use Interaction Protocols to both model formal interactions and ease a cooperative adaptation and guidance of crisis resolution processes. After highlighting the benefits of Interaction Protocols to support this human and collective dimension, the paper presents a protocol meta-model for their specification. It then shows how to suitably integrate specified protocols into crisis resolution processes and how to implement this conceptual framework into a service oriented architecture.

Keywords: Crisis resolution process · Interaction protocols · Process flexibility

1 Introduction

In crisis situations (natural or industrial disasters, explosions of violence …), the various actors driving the crisis resolution have to act immediately and simultaneously in order to reduce its impacts on the real world. To achieve this common goal as quickly and efficiently as possible, these actors (police, military forces, medical organizations …) have to collaborate and act in a coordinated way [1]. The coordination of a crisis resolution process is difficult since it needs to adhere to the evolving requirements of the crisis.

In the framework of the Génépi project[1], we adopt a computer-based approach to support the coordination of actors; it is based on the following requirements:

- A Mediator Information System (MIS, [1]) should be set up within the crisis cell to support the crisis resolution;

[1] Project funded by the French Research National Agency (ANR).

L.M. Camarinha-Matos et al. (Eds.): PRO-VE 2015, IFIP AICT 463, pp. 63–76, 2015.
DOI: 10.1007/978-3-319-24141-8_6

- The partners' Information Systems have to be connected to the MIS in order to declare by means of services, the concrete actions they are able to perform;
- A Crisis Resolution Process (CRP) is to be defined and monitored for orchestrating partners' actions;
- The MIS runs in a Service Oriented Architecture (SOA), which is known for easily dealing with distribution and inter-operability issues;
- Given the dynamic, uncertain and unpredictable actual environment of the crisis, the MIS should also ease the flexibility of the CRP.

In France, a MIS is under the responsibility of a Command and Control Center, called a *crisis cell*, headed either by the local "Préfet" or the Interior Minister, depending on the regional or national scope of the crisis. This crisis cell is composed of representatives of the different public organizations involved in the crisis resolution. The cell, which may be geographically distributed, is in charge of adapting and applying a resolution plan framed by the law and implemented into the Crisis Resolution Process.

According to the Génépi project requirements, the main issue for a crisis cell is to define, maintain and adapt a CRP in a collective way. More precisely, crisis cell actors have to interact with each other (vote, negotiate, delegate, sub-contract …) to guide the CRP execution and adaptation, removing its indeterminism and deciding between the available options. Interactions between the crisis cell members and the actors in the field should also be formalized to guarantee both a good understanding of the orders and a correct interpretation of the feedback information and also to ascribe responsibilities. Consequently, CRPs are interaction-intensive processes. The information, knowledge and decisions shared between the actors are heavily reliant on the interactions between them and they must be logged in order to keep trace of the responsibility of each actor in the crisis management. Interactions in the course of a CRP are almost as important as the coordination of activities. Given all these considerations, the problem addressed in this paper is *how to build and monitor flexible crisis resolution processes, i.e. easily adaptable and integrating a human and collective dimension through interaction support between the actors?*

Flexibility of processes is not a new issue [2, 3]. It is defined in [2] as the "ability to deal with both foreseen and unforeseen changes in the environment in which processes operate". Reference [2] also proposes a taxonomy defining several types of process flexibility along with techniques to support them – flexibility by design, flexibility by deviation, flexibility by under-specification and flexibility by change – techniques which are obviously useful to face the dynamic and unstable context of crises.

However, all the crisis cell members share a CRP and this brings a new requirement. The CRP mode of execution, along with the different alternatives it considers, should be subject to consultation and collective decisions at run-time, since they engage the responsibility of cell members. Hence, in this context, flexibility must also take into account this *human* and *institutional* dimension and integrate interaction places to ease decision-making within the crisis cell.

To address this problem, we propose to coherently combine conventional flexibility techniques with interaction mechanisms into CRPs. These interaction mechanisms are described by means of an Interaction Protocol (IP) [3], defined as a set of structured communication acts, following a recurrent schema, and aiming at

coordinating interventions of involved actors to reach specific objectives. In our context, an IP may correspond to the selection of actors who will execute specific tasks according to their roles, the vote of an important decision between actors, or the negotiation of the service quality of tasks. Consequently, IPs can guide the execution of CRPs and facilitates their adaptation.

Following the typology provided by [11], business process systems can be divided into two classes: Person-to-Person (P2P) process-based systems corresponding to groupware and Person-to-Computer (P2C) process-based systems corresponding to workflow. Our proposition merges these two classes in combing suitably the P2P approach with the P2C one. To this end, the contribution of the paper is threefold: (1) a meta-model for the declarative specification of IPs along with means to integrate these protocols into CRPs; (2) requirements and new functionalities for a workflow-process engine in charge of executing CRPs; (3) a Service Oriented Architecture implementation of our propositions.

The paper is organized as follows. Section 2 discusses CRP flexibility and highlights the benefits of IPs for introducing a human and collective dimension in crisis processes. Section 3 presents a meta-model of protocols and also explains protocol integration into CRPs using conventional flexible techniques. Section 4 defines the requirements that a process/workflow engine has to meet to execute such CRPs. Section 5 presents an implementation of our solution in PEtALs, a specific service-oriented architecture. Finally, Sect. 6 discusses our propositions and concludes the paper.

2 Flexibility of the Crisis Resolution Process

A Crisis Resolution Process describes the coordination of actions performed by actors involved in crisis resolution. A CRP is executed by the Mediator Information System [1] whose aim is to drive the CRP lifecycle: define, adapt, orchestrate and supervise its execution according to the crisis model, which is an accurate representation of the crisis and its evolution. CRP flexibility is supported at two abstraction levels. At the MIS level, CRP flexibility is supported by the dynamicity of its life cycle, while, at the CRP level, it is supported by flexibility techniques both used to adapt the CRP and support formal interactions between the possibly geographically distributed crisis cell members.

2.1 Life Cycle for the Dynamic Management of a Crisis Resolution Process

The process driving the execution of the CRP, called Driving Process (DP), includes a *perception-decision-action* loop. Indeed, it provides means to capture information about the impacted world, to support the definition and the adaptation of the CRP managing the crisis reduction, and to coordinate and assign to each participant the actions to be undertaken. These actions modify the real world and lead to a new iteration, where these tasks are performed again, taking into account the new crisis context. Figure 1 illustrates, through a BPMN diagram, the dynamics of the DP.

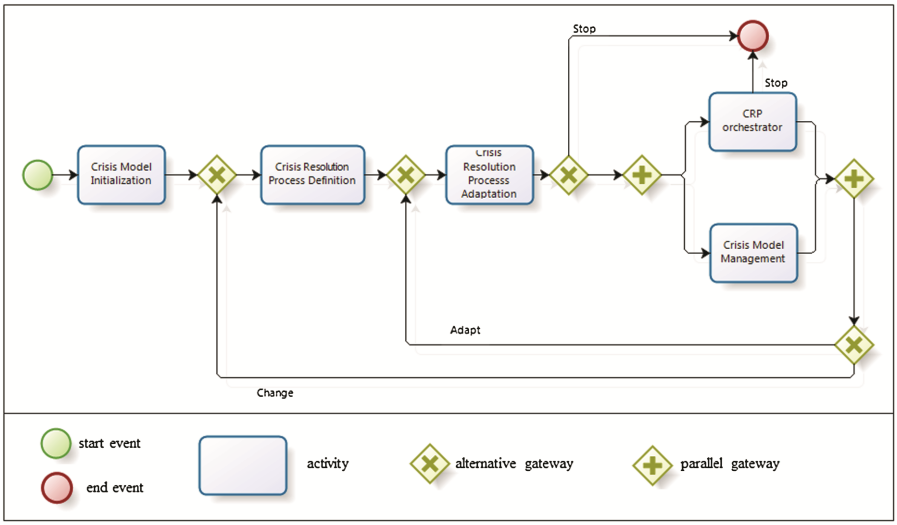

Fig. 1. PMN diagram of the driving process.

The DP structure should be read as follows. Once a crisis cell, responsible for the crisis management, is set up, the Crisis Model Initialization task records into the crisis model the observations of the impacted real world. This crisis model is compliant with a crisis meta-model such as defined in [4].

Using this crisis model, the *Crisis Resolution Process Definition* activity defines the initial CRP. Then, this initial CRP may be adapted in order to introduce human controls into the loop (through interaction protocols) or to take into account specific organizational requirements expressed by actors involved in crisis resolution. More precisely, this adaptation may consist in removing, adding or reordering actions or interaction protocols, or in selecting, among several alternatives, the most relevant one to guide the CRP execution. Then, either the DP stops or continues. In the first case, this means that the crisis is reduced or considered as being solved. In the second case, two activities are concurrently performed: the first one executes of the CRP, while the second updates the crisis model representative of the real world situation. The orchestration of the CRP may be suspended to be either adapted or changed according to the evolution of the crisis model. The *Adapt* case corresponds to a simple update of the CRP and then a new <Orchestration // Crisis Model Management> step is started. The *Change* case corresponds to an evolution of the crisis and a deep modification of the CRP. Consequently, the *Crisis Resolution Process Definition* activity is again performed in order to define a new CRP in line with the new crisis model. Section 4.1 will explain how the *Crisis Model Management* activity impacts CRP *Orchestration*.

The DP structure requires to control the functioning of the orchestrator responsible for the execution of the CRP. This orchestrator must be able to suspend its execution and restart it. Section 4 presents the solutions we provide for that.

2.2 Flexibility Techniques for Adapting the Crisis Resolution Process

Flexibility of a Crisis Resolution Process corresponds to its ability to face changes, i.e. to be adapted in the course of its life cycle. Some of these changes are foreseen and consequently are taken into account when designing the CRP, while others are not and are consequently performed at run-time. We present below appropriate techniques for CRPs flexibility. We distinguish those proposed by the Business Process Community in the context of activity-oriented processes (e.g. [5, 6]), from the one we propose in the paper: interaction mechanisms, involving human groups having to collectively guide the execution of the CRP, remove its indeterminism, discuss its execution options or decide actions to undertake.

Conventional Techniques for CRP Flexibility. Four types of flexibility are identified in [2] and are convenient for CRP. The types of flexibility are: flexibility by *design*, flexibility by *deviation*, flexibility by *under-specification* and flexibility by *change*. Flexibility by design corresponds to foreseen changes in processes that can be modelled when designing them. This type of flexibility depends on the modelling power of the language used for process description. Flexibility by derivation handles, at run-time and at the instance level, unforeseen changes and where the differences with the initial process are minimal. Flexibility by under-specification deals with foreseen changes that cannot be defined at design-time but rather at run-time. This type of flexibility corresponds to the notions of late binding and late modelling [2]. Regarding late binding, a process designer specifies several ways to implement a process or an activity of the process, and postpones the choice of one of these ways until its execution. Late modelling is different from late binding since the designer postpones the way to model the process to its execution. Consequently, the user who will execute the process will have to define it before its execution. Finally, flexibility by change corresponds to unforeseen changes at run-time, which require occasional or permanent modification of process schema. In the context of CRPs adaptation, and as illustrated later, we propose to use flexibility by change and flexibility by under-specification (and more precisely late binding) techniques [5, 6].

Flexibility by Integration of Interaction Protocols. As defended in the introduction, the previous techniques must be supplemented with collaborative facilities in order to integrate interaction places to support crisis cell members' decision-making. Consequently, we propose to use Interaction Protocols [17] to integrate such a dimension into CRPs. In the crisis context, IPs are for instance used to select actors who will execute specific actions (e.g. which hospitals are available for receiving injured people), to organize votes about important decisions between possibly geographically distributed crisis cell members, or to negotiate criteria to perform specific actions. Consequently, IPs within a CRP influences its execution and thus participates to its dynamic guidance and adaptation.

3 Integration of Interaction Protocols into Crisis Resolution Processes

This section first introduces a meta-model of protocols devoted to declarative specification of Interaction Protocols. It then explains protocol integration into CRPs using conventional flexible techniques.

3.1 Declarative Specification of Interaction Protocols

An Interaction Protocol (IP) is defined as a service through its profile and its behaviour. The profile provides all the necessary information for an IP to be found and possibly selected, while the behaviour of an IP defines the operations it performs. As this paper mainly focuses on IP behaviour specification and integration into CRPs, Fig. 2 only introduces concepts for behaviour description, i.e. the concepts of roles, messages exchanged between roles, actions performed within roles, and the way these actions and messages are connected together.

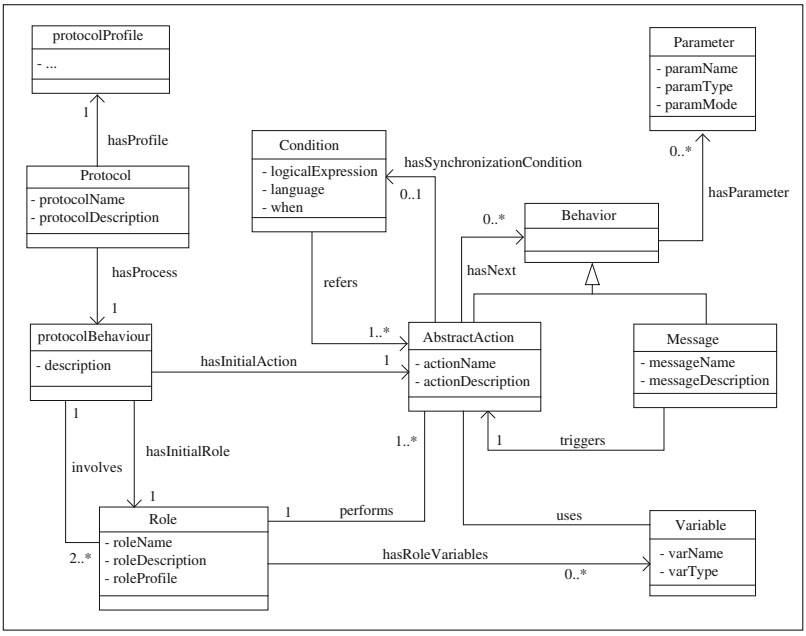

Fig. 2. Protocol meta-model as a UML class diagram.

More precisely, a *protocolBehaviour* has a *description* and involves several roles; it is initiated by one of these roles (*hasInitialRole* association), and it starts triggering an initial action (*hasInitialAction* association). Each *Role* has a name (*roleName* attribute), a description (*roleDescription* attribute), a profile (*hasProfile* attribute), performs actions (*performs* association), and uses some local variables (*hasRoleVariables* association). We distinguish

two kinds of actions for IPs: *AbstractActions*, which correspond to internal actions of roles, and *Messages*, which correspond to messages exchanged between roles. Regarding messages, we specify their name (*messageName* attribute), their description (*messageDescription* attribute) along with the abstract actions (of a role) they trigger (*triggers* association), the local variables they use (*uses* association), and the parameters sent to these abstract actions (*hasParameter* association). Regarding abstract actions, we specify their name (*actionName* attribute), their description (*actionDescription* attribute), and the parameters they receive or send. The code of an action, i.e. the code that will be executed when the action will be performed, will be defined later, at the implementation level (see Sect. 5). We also indicate how actions and messages are connected together (*hasNext* association), along with the synchronization condition when an action follows several previous ones (in order to be able to model join and fork patterns for parallelized actions, if pattern for alternative actions …).

3.2 Flexibility Techniques for Interaction Protocol Integration

Two flexibility techniques among the ones proposed by the Business Process Management community can be used for IP integration. The first technique, flexibility by change, enables the integration of a concrete IP whose profile and behaviour are known at design time. This technique consists in adding an activity into the CRP whose role is to call the IP to be integrated. As illustrated in Fig. 3, we have added the Protocol activity between two conventional activities (Activity-1 and Activity-2) of a CRP.

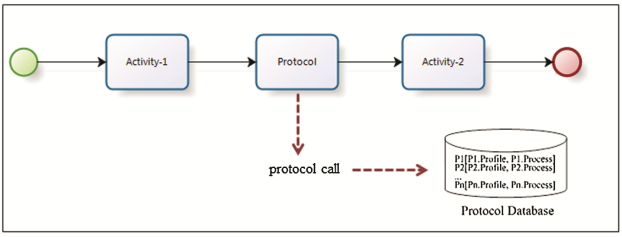

Fig. 3. Flexibility by adding concrete interaction protocols.

As illustrated in Fig. 4, the second technique, combining both flexibility by change and under-specification (late binding) approaches, differs from the previous one since it enables the integration of abstract IPs, for which only the profile is known at design-time. The idea is to postpone the choice of an IP behaviour at run-time. Thus, this technique consists in adding the Protocol Selection activity into the CRP whose role is to select a convenient protocol among the IPs recorded in the Protocol database. When executing this activity, the most convenient protocol, in terms of IP' profile and behaviour is chosen and then executed.

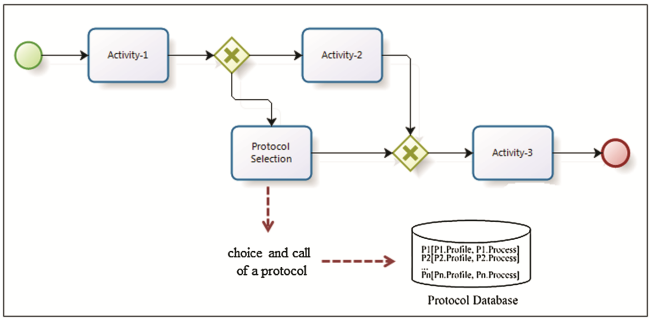

Fig. 4. Flexibility by adding abstract interaction protocols.

4 Crisis Resolution Process Engine Requirements

The two previous techniques may be used both at design-time and run-time. Adding IPs at run time requires that the Process Engine executing the CRP is able to suspend and restart its execution. This section states requirements the Process Engine should meet for this integration.

4.1 Overview of the Process Engine Supervisor

To meet these requirements, we introduce the notions of *checkpoints* and *guidelines* for the Process Engine supervisor. Checkpoints correspond to suspension and observation points of the running CRP, while guidelines correspond to actions allowing modifications of the CRP state (suspend, adapt…) or of its activities states (skip, suspend, allocate…). Figure 5 gives an overview of the process engine supervisor and shows how the Process Engine and the CRP Process Driving components interact.

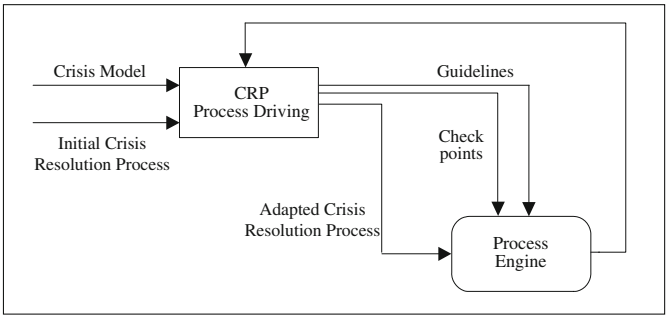

Fig. 5. Process engine supervisor

Let us recall that the Mediator Information System activity consists mainly in CRP definition, adaptation and supervision. The Process Engine performs CRP execution

while supervision is left to the CRP Process Driving component. This component inputs are the initial CRP and the Crisis model (cf. Sect. 2) and its outputs are an eventually adapted CRP (integrating new activities or protocols for instance), execution checkpoints and guidelines for the Process Engine.

4.2 Checkpoints and Guidelines

These two notions support Process Engine supervision. However, we first have to define the states of a CRP and its activities.

States of a CRP and its Activities. Figure 6 presents the different states of a CRP and indicates which transitions are available between them. These states are *defined* (the CRP is defined), *executed* (the CRP is running), *suspended* (the execution of the RCP is suspended in order to adapt or redefine it), *adapted* (the CRP is changed in order to face crisis evolution), or *finished* (the CRP execution is terminated and the crisis is reduced). This state is the final one.

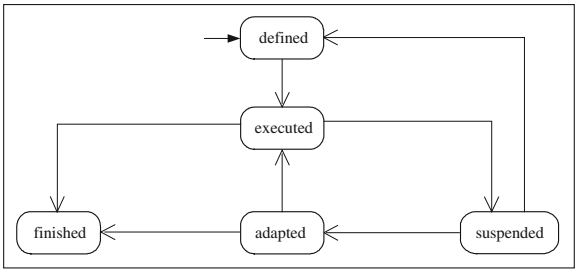

Fig. 6. The CRP states

In the same way, we define the states of CRP activities which have to be performed by actors to reduce the crisis. The different possible states of an activity are: *defined* to indicate that the activity is defined in the CRP, *sensible* to indicate that it will possibly be executed, *allocated* to indicate that it will be executed soon, *executed* for a running activity, *suspended* when its execution is suspended, *finished* to indicate that its execution is terminated, *failed* to indicate that its execution failed, *skipped* to indicate that it will not be executed, and *aborted* to indicate that its execution has been aborted. The possible final states are skipped, finished, failed and aborted. Figure 7 presents the transitions between states.

Checkpoints and Guidelines. Now, we indicate how to supervise the execution of a CRP through the use of checkpoints and guidelines. Guidelines correspond to orders given to the Process Engine to modify the state of the CRP or the state of a CRP activity. Checkpoints correspond to breaks introduced in the CRP. These breaks are associated with activities included in the CRP. When the Process Engine meets a checkpoint, it suspends the execution of the CRP, and the crisis cell may adapt or redefine it by introducing new activities or interaction protocols. Guidelines and checkpoints are expressed as active rules [7] defined as follows:

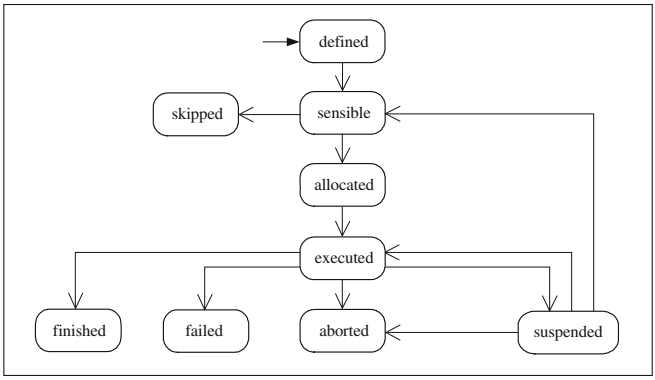

Fig. 7. The states of a CRP activity

Rule <RuleName>
Priority <Number>
When <Event>
If <Condition>
Then <Action>

The Process Engine must be able to interpret these rules. It has to trigger the action part of the rules when the event occurs and the condition is verified. Priority is introduced in order to solve conflicts if several rules may be fired at the same time. Events correspond to temporal events, events from the Crisis model or events raised by the CRP itself (as exceptions in programming languages). Actions associated to checkpoints or guidelines differ. Regarding checkpoints, the triggered action suspends the execution of the CRP; then the crisis cell may analyze the CRP state, redefine it or adapt it by adding new activities or integrating interaction protocols as indicated in Sect. 3.2. Regarding guidelines, the triggered action may modify an activity state or may correspond to notifications sent to the crisis cell.

We can note that this idea of integrating events into processes to make them more flexible is also used in [8] for cross-organizational business processes.

5 A Service Oriented Architecture Implementation

We have implemented our propositions in PEtALs, an open source Service Oriented Architecture on top of an Enterprise Service Bus technology based on JBI (Java Business Integration), JMS (Java Message Service), XSLT (eXtensible Stylesheet Language Transformation) and web services standards (SOAP, WSDL, BPEL).

PEtALs supports mediation and interoperability between software components using adaptors, and provides an engine, called Maestro, for orchestrating BPEL services. In the context of Génépi, PEtALs connects information systems of partners (actors) involved in crisis resolution and orchestrates the CRP. More precisely, partners publish the actions they are able to perform on the impacted world (e.g. stop a fire) as web

services through their information systems. After real world situation analysis, the crisis cell defines a first CRP, which connects the different web services published by partners. This CRP is specified in BPMN and automatically derived in a BPEL process using UML as a pivot model and ATL (Atlas transformation Language) for the derivation [9]. Then, these elements (i.e. the BPEL process and the Maestro engine) are deployed on PEtALs. When running, the CRP triggers services corresponding to either external services such as web services published by partners, web services implementing interaction protocols as BPEL processes, web services supporting the update of the crisis model after collecting information from actors on the field, or internal services such as the Service Retriever component used for finding services. Figure 8 illustrates the connexion between internal and external services within PEtALs.

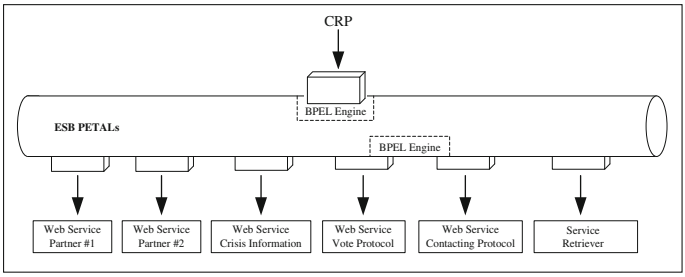

Fig. 8. Conceptual architecture of PEtALs

More precisely, internal and external services are deployed in PEtALs as service units. Each service unit has an endpoint along with its corresponding WSDL (web Service Description Language) specification. An external service is deployed by two service units: the first one for the service itself and the second one for its access through the web (using a SOAP binding component). As indicated above, interaction protocols are mapped onto BPEL processes and encapsulated into web services. Moreover, these BPEL processes are completed with a set of packaged operations that includes the (Java) code concretizing the different abstract actions of protocol roles. Consequently, an IP is deployed in PEtALs by three service units for each of its role: one for the BPEL process corresponding to the considered role, a second one for the operations (Java code) concretizing the different abstract operations of the role, and a third one for accessing the role through the web. Figure 9 illustrates this idea considering a *Contracting Protocol* involving two roles: manager and contractor. Several instances of the *Contractor* role are visualised in this figure.

Regarding the integration of IPs as BEPL process, static integration of IPs (i.e. when defining the CRP) raises no difficulty. On the other hand, dynamic integration of IPs is more difficult and requires being able to suspend the execution of the CP, and then eventually take it up again. The PEtALs Maestro engine supports this functionality. Indeed, this engine uses the results of the work carried out in the context of the Fractal open source project (http://fractal.objectweb.org/), and as any component architecture based on Fractal, Maestro has capacities for introspection.

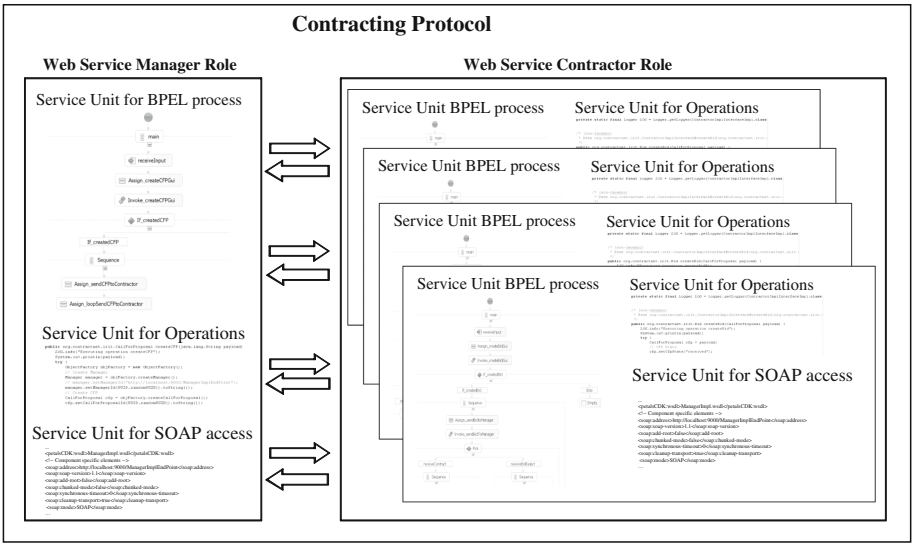

Fig. 9. Modelling IP roles in PEtALs

More precisely, the Maestro engine is composed of two components: (i) a wrapper that translates a BPEL process into a Java class, and (ii) a Process Virtual Machine (PVM) that is responsible for this Java class execution. The process activities included in the Java class are packed as Fractal components. So, thanks to the capability of Maestro to account for the Fractal facilities of the components, users can supervise processes execution, and through predefined or specific controllers, they can drive the CRP using checkpoints and guidelines.

Moreover, Maestro collaborates with a rule engine in order to take into account checkpoints and guidelines. These checkpoints and guidelines are defined as XML rules recorded in a configuration file (rules.xml) and are loaded by the BPEL file of the CRP.

6 Discussion and Conclusion

To deal with the need of a human and collaborative dimension in CRPs, the paper has shown how to integrate interaction places that ease coordination within the crisis cell, the guidance of process execution and collaborative decision-making.

For that purpose, we have defined a Protocol meta-model for the declarative specification of Interaction Protocols (IPs) and have shown how to implement them by means of conventional flexibility techniques (flexibility by change and under-specification [2, 5, 6]). Moreover, the paper proposes new functionalities that a Process Engine should offer to orchestrate flexible CRPs. Finally, to complete this conceptual and platform independent framework, we provide an implementation on a Service Oriented Architecture environment, namely PEtALs. This implementation has required the development of the specific Process Engine Maestro.

The outcomes of IP interaction places that are integrated into a CRP impact and guide the course of the CRP itself. This provides a level of human and collaborative flexibility, which has never been proposed in previous works, while the flexibility of processes is an important issue for the effectiveness of information systems processes [10].

Regarding the state of the art in the Crisis Information System domain, even though a lot of information systems have been developed to support crisis management, most of them are specific to a type of crisis [12], focus on the influence of innovative tools [13], or are limited to simulation or geographic information management and visualization [14]. Collaborative tools (such as CSCW) have also been developed in the Crisis Management domain [15]. Most often they are used just to support interactions among partners and document sharing because such tools do not offer a process management perspective. In this regard, the closest work to ours is [16], developed in the context of the WORKPAD project (http://www.workpad-project.eu). In this work, a P2P architecture is designed from users requirements, and it includes data storage and communication, middleware and user layers. This architecture also manages processes and allows workflow patterns mining. Unlike the architecture proposed in [16], a CRP is a first class citizen component for the supervision, guidance and mastering of a crisis. Finally, our approach remains conceptual and does not impose any technological choice (P2P, Web Services, Grid…), even if this paper reports a SOA-based implementation.

So far, several protocols – a vote protocol, a Matchmaker protocol and an iterative Contract-net protocol – have been implemented. We plan to integrate them and to make them work into the flood of the Loire (longest river in France) CRP, in the context of a simulation exercise involving real crisis cell and actors (firemen, police, medical organizations).

Acknowledgements. Authors thank Nicolas Salatgé, Cecile Faure and Tom Jorquera for their help in the implementation phase of this work. The study was made possible thanks to the financial support of the ANR (French Research National Agency).

References

1. Truptil, S., Bénaben, F., Salatge, N., Hanachi, C., Chapurlat, V., Pignon, J.P., Pingaud, H.: Mediation information system engineering for interoperability support in crisis management. In: I-ESA, pp. 187–197 (2010)
2. Schoneneberg, H., Mans, R., Russell, N., Mulyar, N., van der Aalst, W.: Process flexibility: a survey of contemporary approaches. In: International Workshop on CIAO/EOMAS, at International Conference on Advanced Information Systems, pp. 16–30, Montpellier, France (2008)
3. Rantrua, A., Gleizes, M.P., Hanachi, C.: Flexible and emergent workflows using adaptive agents. In: Proceedings of the 5th International Conference on Computational Collective Intelligence, Technologies and Application, pp. 185–194, Craiova, Romania (2013)
4. Benaben, F., Hanachi, C., Lauras, M., Couget, P., Chapurlat, V.: A Metamodel and its ontology to guide crisis charaterization and its collaborative management. In: International Conference on Information systems for Crisis response and Management, pp. 189–196, Washington, USA (2008)

5. Reichert, M., Dadam, P.: ADEPTflex: supporting dynamic changes of workflow without losing control. Int. J. Intell. Inf. Syst. **10**(2), 93–129 (1998)
6. Adams, M., ter Hofstede, A., Edmond, D., van der Aalst, W.: Worklets: a service-oriented implementation of dynamic flexibility in workflows. In: International Conference on Cooperative Information Systems, pp. 291–306, Montpellier, France (2006)
7. Widom, J., Ceri, S.: Active Database Systems: Triggers and Rules for Advanced Database Processing. Morgan Kaufmann publishers, Burlington (1996)
8. Chakravarty, P., Singh, M.: Incorporating events into cross-organizational business processes. IEEE Int. J. Internet Comput. **12**(2), 46–53 (2008)
9. Trupil, S., Benaben, F., Pingaud, H.: Collaborative process design for mediation information system engineering. In: International Conference on Information System for Crisis Response and Management, Gothenburg, Sweden (2009)
10. Reijers, H.: Workflow flexibility: the forlorn promise. In: International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, pp. 271–272, Manchester, United Kingdom (2006)
11. Ellis, C., Barthelmess, P., Chen, J., Wainer, J.: Person-to-person processes: computer-supported collaborative work. In: Dumas, M., van der Aalst, W., Ter Hofstede, A. (eds.) Process-Aware Information Systems: Bridging People and Software through Process Technology, pp. 37–60. Wiley, Hoboken (2005)
12. de Addams-Moring, R.: Tsunami self-evacuation of a group of western travelers and resulting requirements for multi-hazard early warning. In: International Conference on Information System for Crisis Response and Management, Delft, The Netherlands (2007)
13. Avery-Gomez, E., Turoff, M.: Interoperable communication: an analysis of SMS text-message exchange. In: International Conference on Information System for Crisis Response and Management, Delft, The Netherlands (2007)
14. Dilekli, N., Rashed, T.: Towards a GIS data model for improving the emergency response in the least developing countries: challenges and opportunities. In: International Conference on Information System for Crisis Response and Management, Delft, The Netherlands (2007)
15. Cai, G.: Extending distributed GIS to support geo-collaborative crisis management. Geogr. Inf. Sci. **11**(1), 4–14 (2005)
16. Catarci, T., de Leoni, M., Marrella, A., Mecella, M., Russo, A., Steinmann, R., Bortenschlager, M.: Workpad: process management and geo-collaboration help disaster response. Int. J. Inf. Syst. Crisis Response Manage. (IJISCRAM) **3**(1), 32–49 (2011)
17. Hanachi, C., Sibertin-Blanc, C.: Protocol moderators as active middle-agents in multi-agent systems. Int. J. Auton. Agents Multi-Agent Syst. **8**(3), 131–164 (2004)