Towards Dynamic Workflow Support for Crisis Management

Jan Martin Jansen
Netherlands Defense Academy
jm.jansen.04@nlda.nl

Bas Lijnse
Radboud University Nijmegen
Netherlands Defense Academy
b.lijnse@cs.ru.nl

Rinus Plasmeijer
Radboud University Nijmegen
rinus@cs.ru.nl

ABSTRACT

Current process support technology for crisis management is often limited to either sharing of information or hard-coded process support through dedicated systems. Workflow management systems have the potential to improve crisis response operations by automating coordination aspects. Unfortunately most contemporary systems can only support static workflows, hence yielding inflexible support systems. Recent work on the use of functional programming techniques for workflow modeling has led to the development of the iTask system. It uses function combination to model dynamic data-driven processes and generates executable workflow support systems. Because of its focus on dynamic processes it appears promising for development of flexible crisis response systems. In this paper we present an initial discussion of the potential of the iTask system for crisis management applications. We give an overview of the iTask system, and discuss to what extent it meets the requirements of the crisis management domain.

Keywords

Dynamic Workflow Support, iTask System, Requirements Discussion

INTRODUCTION

Crisis management operations involve cooperation and collaboration between large numbers of diverse organizations (e.g police, firemen, rescue workers, medics). Activities in these operations are highly dynamic and situation dependent. To cooperate and collaborate, activities by diverse organizations must be synchronized (or at least deconflicted) with one another. At first glance, Workflow Management Systems appear to have potential to support. WFMSs are computer applications that coordinate, generate, and monitor tasks to be performed by human workers and computers. Every activity in a crisis management operation can be considered a task. Activities can depend on each other and must be performed in sequence, while other activities may be carried out in parallel. The workflow system can be used to support the distribution and monitoring of these activities. But there are some serious problems, as already acknowledged by Fahland and Woith (2008) and Sell and Braun (2009). First, contemporary workflow systems are commonly rather rigid because they only model the static flow of control. Second, the activities to be conducted for tackling a crisis often cannot be captured in a predefined plan. Only a rough sketch of the actions to be taken can be given. Plans can be further refined only at runtime, when more information becomes available. Most workflow systems cannot deal with this.

Recent work on the use of functional programming techniques for workflow modeling has led to the development of the iTask system (Plasmeijer, Achten and Koopman 2007). The iTask system is a domain specific workflow language embedded in the functional programming language Clean, enabling the creation of data-driven dynamic workflow systems. It supports data dependent behavior of tasks, where the new tasks to do may depend on the results of previous tasks. The iTask system also allows for on-the-fly adaptation of tasks.

Reviewing Statement: This paper represents work in progress, an issue for discussion, a case study, best practice or other matters of interest and has been reviewed for clarity, relevance and significance.

THE ITASK SYSTEM

The iTask system (itasks.cs.ru.nl) is a domain specific workflow language embedded in the functional programming language Clean. It enables the creation of dynamic workflow systems. In the iTask system a workflow consists of a combination of tasks to be performed by humans and/or automated processes. From iTask specifications complete web-based workflow applications are generated. The system is based on open web-standards and can be accessed by anyone who has access to Internet, including many mobile devices.

The iTask system is built upon a few simple concepts. The main concept is that of a typed task. A task is a unit of work to be performed by a worker or computer (or a combination of both) that produces a result of a certain type. A task can be a single (black box) step, or a composition of other tasks. The result of one task can be used as the input for subsequent tasks, and therefore these new tasks are dynamically dependent on this result. iTask allows for the data dependent sequential and parallel execution of tasks where information is automatically transported between tasks. Result types are not limited to simple data such as integers, records, etc., but can also be documents, or even new tasks.

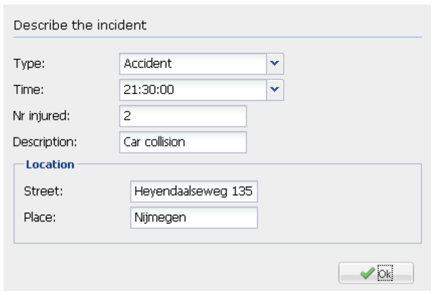
Programming Workflows

The iTask standard library offers several functions for creating basic units of work. An important example is the generic task where a user is asked to supply information. The generation of a web-form to enter information and the processing of its result are handled fully automatically by the system. In this way data entry tasks are created in just a single line of code. Figure. 1 shows the code and the generated form for an `Incident` data type. This code comes from an example application that dynamically allocates ambulances from multiple ambulance posts based on location, number of injured and availability in case of an incident.

```
:: Incident = { type :: IncidentType
              , time :: Time
              , nrInjured :: Int
              , description :: String
              , location :: Location
              }

:: IncidentType = Accident | Fire | Fight | Other String
:: Location     = {street::String, place::String}

enterIncident :: Task Incident
enterIncident = enterInformation "Describe the incident"
```



Describe the incident

Type: Accident
Time: 21:30:00
Nr injured: 2
Description: Car collision
Location
Street: Heyendaalseweg 135
Place: Nijmegen

Figure 1. A Generic Data Entry Task for Incident Data

An obvious advantage of such compact definition of data entry tasks is that it enables readable and easily modifiable workflow specifications. But there are some less obvious, but more important ones: First, the separation of declarative task definition and generic implementation enables different implementations for different devices. Second, because interfaces can be automatically generated, the system can automatically provide a fall back based on manual data entry for every task. Even for tasks that were designed to receive their input through an automated process.

Other examples of basic task functions are: listing all users of the system (if necessary grouped by their role); tasks that return at a predefined moment in time or after an amount of time; tasks that communicate with other applications or web services (for the exchange of information).

New tasks can be composed from other tasks by using combinator functions. We distinguish between combinators that say something about the order in which tasks have to be performed and combinators that say something about an individual task: who has to perform it; where to store information about the task, etc.

In contrast to most workflow specification languages, information is passed explicitly from one task to another in the iTask system. In a *sequential* composition of two tasks, the first task is activated first and when it finishes, the result is passed to a second task, which takes this result as its input. In code, this is denoted by:

```
first_task >>= second_task_function
```

`t>>=f` (or t followed by f) integrates computation and sequential ordering in a single pattern. In this way the second task can dynamically adapt to the result of the first task. In other workflow formalisms it is harder to specify a function that acts on the result of a preceding task because only control is passed between tasks.

The `parallel` combinator can be used for executing tasks in parallel. It can be parameterized with predicates such that many patterns of parallel composition can be expressed using this single combinator. For example: or-, and- and ad-hoc (conditional) parallelism.

Dealing with Dynamic Behavior: Exceptions and Change

Several authors (Sell and Braun, 2009; Fahland and Woith, 2008) already indicated that workflows need to be adaptive to be of use for crisis management operations. iTask offers three constructs to achieve this:

The sequence (>>=) combinator is used to make tasks dynamically dependent on results of previous tasks.

The iTask *exception* mechanism can be used in case the normal course of actions is affected, and a new procedure should be started. A task may throw an exception in case an exceptional situation occurs. The entire workflow the task is part of is now stopped (if there are parallel tasks in it, the users participating in these tasks are informed). The exception is passed to an exception handler that can start a new task using information raised in the exception. Exceptions enable the separation of uncommon borderline cases from the regular workflow.

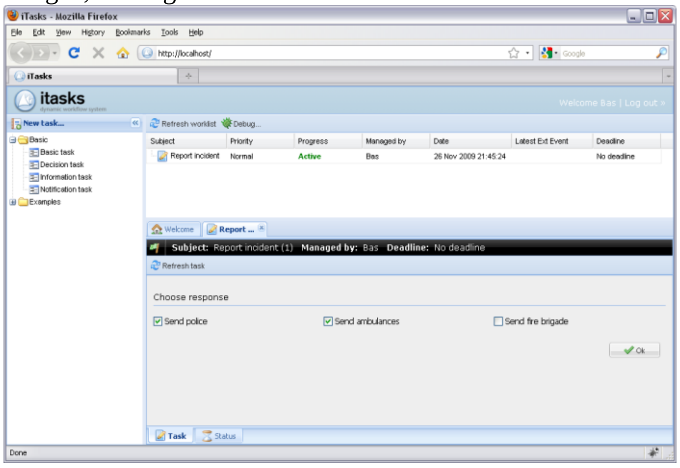
The *change* concept is complementary to that of the exception. A change is something that is triggered from outside the specified workflow. Tasks on which people are working can be replaced on-the-fly with other tasks. An example of a change is the replacement of a complex process by a simple to-do list, in case the user has determined that the process is inappropriate for the current situation, or the replacement of a task by manual entry of a result that is obtained outside the workflow system.

Working on Tasks

When a workflow specification is compiled, a server executable is generated that coordinates tasks through a set of web services. It serves task lists, a workflow catalogue, and high level user interface definitions of concrete tasks. Users can access these services with a generic web based application (Figure 2).

Tasks that a user needs to perform are presented in the task list *inbox* displayed in the upper right pane. The state of a selected task is displayed in the lower right task pane. Tasks can be selected in any order, or simultaneously, allowing a user to determine a preferred order of execution. All work is immediately synced with the server.

Users can start new workflows by selecting them in the left workflow pane. In general any number of workflows can be started.



Subject	Priority	Progress	Managed by	Date	Latest Ext Event	Deadline
Report incident	Normal	Active	Bas	26 Nov 2009 21:45:24		No deadline

Subject: Report incident (1) Managed by: Bas Deadline: No deadline

Choose response

[✓] Send police [✓] Send ambulances [ ] Send fire brigade

Figure 2. A Screenshot of the iTask Client Application

ITASKS FOR CRISIS MANAGEMENT?

Although the iTask system has not specifically been designed with crisis management applications in mind, we contend that it is a potentially valuable tool for building crisis management systems. To find out what is demanded from crisis management systems, such that we can identify challenges for further development, we turned to the literature for requirements. Unfortunately, different authors use different requirements when proposing technology for this domain (e.g. Ingmarsson, Eriksson and Hallberg, 2009; Sell and Braun, 2009). What we needed was an independently defined set of general requirements. Jul in (2007) provides such a set in the form of five design requirements distilled from an analysis of the crisis domain in general. In this chapter we discuss the strengths and weaknesses of the iTask system in light of each of these requirements:

- **Design Requirement #1:** Response technology should seek to support just-in-time learning, first, of the task the tool is intended to support, second, of the needs and goals of the present operation, and, third, of disaster management practices in general.
- **Design Requirement #2:** Response technology, even when focused on agent-driven tasks, should seek to aid response-driven tasks, such as planning, coordination and resource management.

- **Design Requirement #3:** All response technology should actively nurture cooperation, collaboration and partnership formation.
- **Design Requirement #4:** Response technology, while imposing standard structures and procedures, must, insofar as possible, allow flexibility and deviation in their application.
- **Design Requirement #5:** Response technology should aim for graceful augmentation, allowing the technology to be integrated in or removed from the user's activities with a minimum of disruption.

Requirement #1: Just-in-time Learning

Because people do not need to know what they will have to do in advance, the step-by-step guidance through standard procedures by a workflow system is essentially just-in-time learning of those procedures. The workflow specification guides people through procedures they might have never done before. Once users learn how to use the interface to find out what tasks they have to do, and how they can select tasks to work on, they can rely on the system to tell them what needs to be done. To ease the initial learning curve, the iTask user interface has been designed to resemble an e-mail client as much as possible. Users can simply think of the system as a special e-mail system where all messages in their inbox happen to be requests to do something.

A weakness of the iTask system is that the goals and instructions of tasks are communicated primarily through text as defined in the workflow models. When a user is presented with a task having instructions he or she cannot understand, or even worse, can misunderstand, there are no built-in ways to easily resolve that knowledge gap. The learnability of the tasks is therefore almost completely determined by the degree to which the workflow models supply enough information. Of course, this problem also exists for paper handbooks and contingency plans. Interactive workflow systems have an opportunity to do more, e.g. to provide access to information sources, or to provide easy communication to ask peers help.

Requirement #2: Response Driven Tasks

A workflow system, by definition, supports response driven tasks, since its sole purpose is to automate the coordination and execution of standard procedures. It has the additional advantage over hard-coded support systems of having inspectable models that, at run-time, can be queried to get information about what is going on. The dynamic data-driven workflow models that are used by the iTask system have the additional potential of enabling flexible resource allocation and planning. Data that becomes available as a result of performed tasks can be used for the (re)distribution of resources or for planning/scheduling of other tasks. However, currently available resource allocation combinators in the iTask system's standard library are purely algorithmic. It is possible to integrate stochastic or other predictive models to distribute tasks and resources, or to support decision making at crucial points in a workflow. Having such tasks available in a library of the workflow language could further improve the support of response driven tasks.

Requirement #3: Cooperation and Collaboration

Cooperation and collaboration are supported in iTask workflow models by (re) assigning tasks to users and routing the task results from one user to another. Tasks can be delegated and tracked. It is also possible to define workflows that add new users to the system, who then immediately can get tasks assigned to them.

The multi-user features of the iTask system make it possible to define workflow models that involve multiple users. However, to assume that therefore it *"nurtures cooperation, collaboration and partnership formation"* would be too shortsighted. There are still many things that should be facilitated to promote cooperation, regardless of the concrete tasks at hand, such as for example, integrated communication capabilities (chat, voice, video) to enable users to discuss the tasks they are working on, or formation of ad-hoc teams of users.

A more fundamental challenge will be a shift to multi-user tasks. Currently, tasks are always assigned to, and managed by, a single individual. Relations, both formal and informal, between users are not modeled in the iTask system. In daily life, however, it is not uncommon to work together on a task without exactly dividing it into discrete subtasks, or to have shared responsibility for a task. When concurrently working on tasks by multiple users is possible, one could leverage the fact that tasks can produce new tasks, to cooperatively define and execute a workflow with a group of people.

Requirement #4: Flexibility

Flexibility is a feature of the iTask system that pointed us to the potential usefulness of dynamic workflows for crisis management in the first place. Because iTask workflow specifications support the modeling of dynamic processes at multiple levels, it is potentially capable of complete compliance with this fourth requirement. However, although it is technically possible to define very flexible workflow models, the usefulness of this expressive power is constrained by the interface through which it is exposed to end-users. An important research challenge will be to develop generically applicable problem solving patterns that can be applied when normal procedures do not apply. More research is also needed on what information is required by users to become aware that there is a need for deviation from standard procedures, and what is required to decide what course of action is to be selected to resolve the issue.

Requirement #5: Graceful Augmentation

Removal of a workflow system during the execution of an operation guided by it, will always cause disruption. However, it is possible to meet this requirement as closely as possible by reducing the amount of disruption if (a part of) the system is temporarily removed. The current iTask system does not specifically address this issue, because network infrastructure has been assumed to be available. However, it has been shown that it is possible to run parts of workflows offline (Plasmeijer et al, 2008), by transferring part of the workflow computation to the client system. A last resort would be to use the system in a controlled environment such as a command post while communicating tasks through other channels. In this case it could still have advantages over written handbooks, because iTask workflow specifications are active and dynamic.

CONCLUSIONS

In this paper we presented dynamic workflow modeling, as implemented in the iTask system, as a candidate platform for developing applications to support crisis management operations. Although this system has not been designed for crisis management, we contend that, because of its unique features like: data driven, parameterizable workflows and extensive support of dynamic behavior, it has potential value in this domain. We have explored this potential through a discussion of iTasks strengths and weaknesses in light of five key design requirements for crisis response technology defined by Jul in (2007) . The aim of this discussion has been to identify challenges for further research, which most notably are: collaboration and effective use of flexible workflows. By addressing these challenges, we hope to develop the iTask system into a valuable tool for building systems that flexibly support people under demanding circumstances.

REFERENCES

1. Fahland, D. and Woith, H. (2008) Towards process models for disaster response. In Leoni, M. de, Dustdar, S. and Hofstede A, editors, *Proceedings of the First International Workshop on Process Management for Highly Dynamic and Pervasive Scenarios (PM4HDPS),* co-located with *6th International Conference on Business Process Management (BPM'08)*.
2. Ingmarsson, M. , Eriksson, H. and Hallberg, N. (2009) Exploring development of service- oriented C2 systems for emergency response. In Landgren, J. and Jul, S., editors, *Proceedings of the 6th International ISCRAM Conference*, Gothenburg, Sweden.
3. Jul, S. (2007) Who's really on first? A domain-level user, task and context analysis for response technology, In Walle, B. van de, Burghardt, P. and Nieuwenhuis, C., editors, *Proceedings of the 5th International ISCRAM Conference*, Delft, the Netherlands.
4. Plasmeijer, R., Achten, P. and Koopman, P. (2007) iTasks: executable specifications of interactive work flow systems for the web, In *Proceedings of the 12th International Conference on Functional Programming, ICFP'07*, pages 141–152, Freiburg, Germany, ACM Press.
5. Plasmeijer, R., Jansen, J., Koopman, P. and Achten, P. (2008) Declarative Ajax and client side evaluation of workflows using iTasks, In *Proceedings of the 10th International Conference on Principles and Practice of Declarative Programming, PPDP'08*, pages 56–66, Valencia, Spain.
6. Sell, C. and Braun, I. (2009) Using a workflow management system to manage emergency plans, In Landgren, J. and Jul, S., editors, *Proceedings of the 6th International ISCRAM Conference*, Gothenburg, Sweden.