Capturing the Netherlands Coast Guard's SAR Workflow with iTasks

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

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

Ruud Nanne
Netherlands Defense Academy
ram.nanne@nlda.nl

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

ABSTRACT

The dynamic nature of crisis response operations and the rigidity of workflow modelling languages are two things that do not go well together. A recent alternative approach to workflow management systems that allows for more flexibility is the iTask system. It uses an embedded functional language for the specification of workflow models that integrates control-flow with data-flow in dynamic data-dependent workflow specifications. Although there is a variety of publications about the iTask workflow definition language (WDL) and its implementation, its applications have been limited to academic toy examples. To explore the iTasks WDL for crisis response applications, we have developed an iTask specification of the Search And Rescue (SAR) workflow of the Netherlands Coast Guard. In this specification we capture the mix of standard procedures and creative improvisation of which a SAR operation exists.

Keywords

Workflow Specification, Search and Rescue, Coast Guard.

INTRODUCTION

Workflow management systems (WFMS) are not particularly well known for their flexibility. Hence, the thought of using a WFMS to support the coordination of Coast Guard Search and Rescue (SAR) operations may seem to be a doomed endeavour from its onset. Many contemporary WFMSs use a flow-diagram graphical language interpreted by a workflow engine to orchestrate or dictate the work to be done. The implicit assumption of such specification languages is that all activities and the order in which they will be executed can be specified in advance. That this assumption does not hold for SAR operations at the Netherlands Coast Guard is best illustrated by the following quote from the OPPLAN-SAR, their primary operational procedure document (freely translated from Dutch):

> *"Because we know from experience that no two SAR incidents are alike, it is impossible to define a fixed, extensive, always applicable procedure"*

Jansen et al. (Jansen, Lijnse and Plasmeijer 2010) claim that, by contrast to other WFMSs, the iTask system's approach to workflow specification is expressive enough to capture the dynamic nature of tasks required for crisis management. To date, this claim has not been fully tested. Although the iTask system has been subject of extensive research in the field of programming language design (Plasmeijer, Koopman and Achten, 2007; Jansen, Plasmeijer, Koopman and Achten, 2010), no work exists that explores its applicability outside the realm of academic toys.

Reviewing Statement: <to be completed by the editors> This paper has been fully double blind peer reviewed./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 purpose of this study is to explore the strengths, weaknesses, and other properties of the iTask workflow definition language (WDL) when it is used to specify a real-world crisis response workflow. The Netherlands Coast Guard manages small and large crises on a daily basis. Their SAR operations provide an interesting case, because they consist of a mix of following standard procedures and ad-hoc crisis management. It is also a convenient case to study because their procedures are well documented and all incidents are logged.

The iTask System

The iTask system (iTasks) is a workflow language embedded in the functional programming language Clean (clean.cs.ru.nl). 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 applications are based on open web-standards and can be accessed by anyone who has access to Internet, including many mobile devices. iTasks is a textual formalism (i.e., a programming language) and offers a much higher degree of flexibility than graphical formalisms that are in use for specifying workflow systems. The iTask system is built upon a few core 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 depend dynamically on this result. iTasks allows sequential and parallel execution of tasks, with information automatically being transported between tasks. Intermediate results of tasks that are executed in parallel can be used to decide whether the execution of other tasks running in parallel should be stopped or altered. Result types are not limited to simple data such as integers, records, etc., but can also be documents, or even new tasks. Tasks can be explicitly allocated to persons, and dynamically reallocated if necessary.

Two concepts especially contribute to the language's expressiveness. The first is that tasks can be higher order. This means that the result of a task can be a new task. For example, a task may use the output of one or several other tasks to construct a new set of tasks and the way they must be executed: sequentially, in parallel, or a combination of both. As a consequence, the workflow specification cannot be completely determined beforehand, but is constructed iteratively during execution. The second is that tasks can be parameterized by data types. This makes it possible to define generic tasks or task structures that are independent of the specific result type of the task. This allows them to be used in multiple contexts.

The iTask implementation is a research prototype, because its WDL is still evolving. Development of the WDL focuses on the exploration of workflow specification concepts for applications in dynamic domains, such as crisis management, command & control, and medical support systems. The prototype status means that the core WDL concepts are available, and workflow specifications can be compiled to complete executable workflow support systems. However, there is no large standard library as could be expected from a production WFMS. Features without scientific interest are added by demand.

The Netherlands Coast Guard

The Netherlands Coast Guard is an independent civil organization with its own responsibilities and competences. The Coast Guard functions as a central reporting, information, and coordination centre in its role as the National Maritime and Aeronautical Rescue Centre (Joint RCC). Its main area of operations is the North Sea. This is one of the busier shipping routes in the world, populated with a crowded combination of commercial and private vessels. Resources must be deployed at short notice when an accident or incident at sea occurs. These resources consist of vessels, airplanes, helicopters, and rescue team stations. Each has different sponsors, different lines of communication, and different procedures, making the communication and coordination of crisis response complex.

One of the main responsibilities of the Coast Guard is execution of the SAR service. This service is responsible for searching for aircraft, ships, and oil drilling platforms in distress within the North Sea and in Dutch coastal waters and for rescuing their crews and passengers.

The Coast Guard currently uses a variety of communication systems (radio, telephone, telex, etc.) and systems for information sharing, for information retrieval (databases, documentation), and for information logging. Logged information is used both for information sharing during operations and for evaluation afterwards. The Coast Guard's

current systems offer only modest workflow support in the form of simple action plans, digital procedure documentation, and predefined forms.

LITERATURE REVIEW

Several authors discuss the use of WFMSs to handle crisis response operations. They all recognize that workflow systems have the potential to offer better support than the use of printed document procedures only. They agree that adaptability is a key issue to be solved if these systems are to be really useful for Crisis Response Operations.

Sell and Braun (2009) defined a number of generic requirements that a WFMS should fulfil in order to be useful for crisis response operations. According to them, the WFMS should:

1. support the management of resources;
2. always depict the current state of deployment;
3. allow the adaptation of the workflow before and during execution;
4. support the delegation of measures;
5. support the execution of workflows.

Based on these requirements, Sell and Braun propose a workflow data model that fulfils these requirements, but do not give an implementation of it. Jansen, Lijnse, and Plasmeijer (2010) claim that iTasks provides concepts that are powerful enough to fulfill these requirements. In this paper we focus on requirements 2 and 3.

Fahland and Woith (2008) also focus on the use of WFMSs for crisis management. They observe that routine processes, even if specifically designed for a situation, should never be enacted blindly. Rather, actions and processes should adapt their behaviour based on observations and available information. They propose specifying an adaptive process as a set of scenarios using Petri nets. Their operational model provides an adaptation operator that synthesizes and adapts system behaviour at run-time, based on these scenarios.

Peukert, Lincourt, and Zimmermann (2009) developed the Collaborative Task Manager as a tool to model and execute workflows. This system enables modelling the exchange and reuse of user-defined task structures. It uses an email-based system for the distribution and delegation of tasks. It supports ad-hoc deviations from pre-defined plans. Due to the tool's tracking functionality, previous ad-hoc processes can be analyzed on the back-end system to give guidance to the current process and to discover best practices.

RESEARCH QUESTIONS

Since the iTask system has not yet been applied in real-world applications, our research goal is to address a question that extends beyond the specific Coast Guard case. We wish to know if the iTask specification language can be used to specify workflow support systems for real-world crisis response operations, and if not, what is lacking.

We contribute to the broader question by developing a specification for a case that is both real-world and dynamic. Coordinating SAR operations provides us with a reference case, enabling us to answer the following research questions:

RQ1: What are the properties of the specification?

> How is it structured and why? Which features of the iTask WDL are used?

RQ2: Is the iTask WDL expressive enough for this case?

> Are there aspects that could not be specified using the WDL's primitives? If so, is there a fundamental reason why not? Or can the WDL be extended to include these aspects? Are there aspects that could only be specified using the WDL's dynamic features?

The primary purpose of answering these questions is to better understand the strengths and weaknesses of the iTask WDL as a method for capturing crisis response workflows. The effectiveness of execution of these workflows and the quality of the support systems generated from them is beyond the scope of this paper.

As a bonus, the specification also gives us an insight into which aspects of the Coast Guard's SAR work would benefit from further automation, and - perhaps just as importantly - which aspects should be left to skilled operators.

METHODOLOGY

To answer our research questions, we have conducted a qualitative explorative case study using document review complemented with observation and in-situ interviews. This specification is primarily based on information found in the following reviewed documents:

- The OPPLAN-SAR V7, which is the primary operational plan containing high-level procedures, contracts between involved authorities, other agreements, and background information.
- Internal operational procedure documents. These contain more detailed guidelines than described in the OPPLAN-SAR and can be considered as its operational implementation.
- Configuration databases from VISION, the Coast Guard's current logging and incident management system. These databases provide insight into what information is collected during incidents.
- The Netherlands Coast Guard public website (www.kustwacht.nl).

We searched these documents for fragments containing procedure descriptions and compiled them into a single file. These fragments were then formalized using the WDL of iTasks release 10.8.

The documents are not followed blindly, but are used as guidelines. Well-trained officers have their individual interpretations of the procedures. Hence, we made four on-site visits to the command centre. The first visit was a guided tour of the organization, and included an in-depth demonstration of the communication and information systems currently in use. The other three visits were devoted to following an operational team during the course of a shift to observe the actual workflow of incident coordination and to interviewing the duty and watch officers in-situ during quiet moments. These interviews focused primarily on storytelling to give context to the documents.

Based on the increased understanding of the domain from these visits, another pass over the documents was made, and the fragments were integrated into a single specification.

RESULTS

Properties of the Specification

Because of the size of the specification (± 2700 LOC), and because it contains proprietary information, it is impossible to cover it in full detail here. Instead, we present an overview of the specification, explaining the key parts and their relations, illustrated with examples taken from the specification. Details of the iTasks WDL can be found in (Jansen, Plasmeijer, Koopman & Achten, 2010).

Overview

At first glance, the process of managing a SAR operation appears to be simple. When a distress call is received, actions are taken immediately to collect more information and to assess the situation. This may vary from a simple request for medical advice to a full-scale disaster; in each case, the set of connected events is known as an incident. When the situation requires, rescue vessels and aircraft are dispatched to search for the originator of the distress call and to rescue any crew or passengers. The incident is over when the vessel or people have been located and rescued or when no reasonable hope of rescue exists. In real incidents, managing a SAR operation is a complex, highly parallel process consisting of many interdependent actions, communications, and decisions, all based on incomplete and uncertain information.

The iTasks SAR workflow specification consists of two main parts. The first part defines the tasks for responding to inbound communications. When a Coast Guard officer answers the telephone or receives a radio call, he/she does not know the topic of the conversation in advance. Any call can be related to an ongoing incident, or it can initiate a new one. Therefore, responding to inbound communication is specified separately from subsequent actions. The second part specifies the coordination of response actions once a new incident has been initiated.

Incident coordination again splits into two parts: information management and action coordination. The first part specifies the type of information that is collected, stored, and viewed to assess the situation and make decisions. The second part deals with the definition, planning, execution, and monitoring of the actions taken.

Part 1: Inbound Communication

With respect to SAR, the Netherlands Coast Guard is primarily a reactive organization. Although precautionary actions are taken, such as the strategic positioning of rescue vessels during storms, the normal mode of operation is that nothing is done until a distress call or other request for assistance is received.

Calls for help or other reports of potential incidents can come at any time and through a variety of communication systems. Most incidents are initially reported via the emergency VHF channel 16 or through a regular telephone call, but incidents can also be triggered when a message from a variety of (semi-automated) alarm systems is received. Using the internationally standardized GMDSS (Global Maritime Distress and Safety System) equipment, ships may broadcast emergency messages through a number of different channels. GMDSS distress messages can be received on Digital Selective Calling (DSC) radio or via INMARSAT satellite telephony. Additionally, incidents can also be reported by email or reported through a data-feed from the dispatch centre of the police and rescue services.

Depending on the communication system through which the inbound communication is initiated, different information is available. A message from an EPIRB (an automated radio beacon) contains specific identification and position information, while no a-priori information is known for a voice call on VHF Channel 16. To take this difference into account, we have specified a specific response task for each type of communication system. Which a-priori information is available is reflected in their signatures:

```
respondVHFChan16    ::                          Task (Maybe Incident)
respondPhoneCall    :: (Maybe PhoneNumber)    → Task (Maybe Incident)
respondINMARSATCall :: (Maybe INMARSATNumber) → Task (Maybe Incident)

respondDSCMessage   :: DSCMessage             → Task (Maybe Incident)
respondCS406MHz     :: CS406MHzMessage        → Task (Maybe Incident)
respondINMARSATC    :: INMARSATCMessage       → Task (Maybe Incident)
respondEmail        :: EmailMessage           → Task (Maybe Incident)
```

These tasks address the handling of one message or conversation. They define the task of determining whether it is related to an ongoing incident, whether a new incident should be created, or if it can be safely ignored (e.g., it is a non-operational phone call). The `Maybe` annotations in the signature indicate that the creation of an incident is optional. Arrows denote computation. These inbound response tasks are tailored to the main communication systems, but information could also be received from unspecified sources via the public telephone network. For example, suppose that the friend of a watch officer gets into difficulties while sailing and calls the watch officer on his/her personal cell phone. For such cases, there is also a task defining the ad-hoc creation of a new incident:

```
createNewIncident   :: Task Incident
```

Part 2: Incident Coordination

Once an incident has been reported via one of the inbound communication channels it is the responsibility of the duty officer to coordinate a response operation. This involves taking a mixture of actions that are prescribed by standard procedures and ad-hoc actions guided by the duty officer's continuous reassessment of the situation. In the iTask specification we have modelled the task of coordinating an incident as follows:

```
coordinateIncident :: Incident → Task Incident
coordinateIncident incident
 =   (manageInformation incident -||- coordinateActions incident)
 >>= postIncidentActions
```

This states that the task of coordinating an incident consists of two steps in sequence (expressed by the `>>=` operator). In the first step, two tasks are executed in parallel: `manageInformation` and `coordinateActions` (combined using the `-||-` operator). Then, when the operation has been completed, a task `postIncidentActions` is executed. The `manageInformation` task states that the duty officer makes sense of the situation by gathering together and assessing the available information. At the same time, actions are taken by the duty and watch officers either to collect more information, to distribute information to others, or to instruct and command the rescue units. The aggregation of all these actions is captured by the `coordinateActions` task. Everything that has to be done after an incident, such as writing reports or evaluating the incident, is captured by the `postIncidentActions` task.

Part 2a: Information Management

Although information management is a key task during SAR operations, its primary specification, the data model, is outside the scope of a WDL. From a workflow perspective, the browsing, viewing, and editing of information about an incident can be viewed as a single task. In the specification, we have defined this task only minimally for testing purposes using iTasks' built-in object database tasks. However, because the data underlying the `manageInformation` task is accessed and modified by the `coordinateActions` part of the specification, an interface to the available information is needed. This interface is specified as a collection of data types that define incident-related data. For example, there is a type `Incident` that represents the collection of all known information about an incident, and there is a type `Contact` that contains information about a party involved in the incident, such as a ship or an aircraft. The definition of these types is given below:

```
:: Incident =
  { incidentNo :: IncidentNo
  , title      :: U String
  , summary    :: U Note
  , type       :: U IncidentType
  , phase      :: U EmergencyPhase
  , weather    :: U WeatherInfo
  , contacts   :: [Contact]
  , log        :: [LogEntry]
  , closed     :: Bool
  }
```

```
:: Contact =
  { contactNo        :: ContactNo
  , type             :: U ContactType
  , name             :: U String
  , position         :: U GeoPosition
  , contactOn        :: U ContactMedium
  , isCoastGuard     :: U Bool
  , inDistress       :: U Bool
  , canHelp          :: U Bool
  , reportedIncident :: U Bool
  , notes            :: U Note
  }
```

Because information is likely to be incomplete or uncertain, some data is wrapped in a parameterized type `U`:

```
:: U a = Unknown | Known (a,Timestamp,Source) [(a,Timestamp,Source)]
```

This defines that values can be either unknown or known. For known values, the source from which the information came and the time that it became known are tracked. A list, expressed in code by `[]` brackets, of previous values is maintained to log changes over time.

Part 2b: Action Coordination

The crux of the specification is the definition of the `coordinateActions` task. Based on interviews with officers and the reviewed documents, we established that the specification of the coordination workflow needs to comply with a set of constraining principles:

- The duty officer coordinates the operation, not the WFMS.
- Actions prescribed by standard procedures do not apply to every incident.
- The order in which actions are taken is not fixed, but subject to the duty officer's judgment.
- Ad-hoc actions defined during an operation are necessary to supplement standard actions.

From these principles, we conclude that a rigid specification of predefined tasks executed in a fixed order, as is common in workflow specifications, is not an option. We need a formalization that retains more flexibility. We achieved this by exploiting the fact that the iTasks WDL is embedded in a pure functional language. We capture tasks that are executed in this context by a special data type `HSTask` (Hierarchical State Task) and use type abstraction and higher order functions to define an abstract recipe to compute concrete tasks from values of this type. The `HSTask` data type is defined as follows:

```
:: HSTask s =
 { meta       :: HSMeta
 , activity   :: HSActivity s
 , refinement :: HSRefinement s
 }
```

`HSTasks` specify a task in three parts. The `meta` part specifies meta-data that summarizes the activity to make it possible to choose which activities to start, the `activity` part defines how the task is to be completed, and the `refinement` part defines how to refine a task into smaller sub-tasks. The `HSTask` type is parameterized with the parameter `s` that defines the type of information that is available to complete the task. For the tasks in the SAR specification `s` is `Incident`. This means that all information about a specific incident is available during the activity.

```
:: HSMeta =
 { name        :: String
 , title       :: String
 , description :: Note
 }
```

The meta-data is straightforward except for the distinction between name and title. The name of the `HSTask` is a unique identifier that makes it possible to keep track of tasks that should be executed only once during an incident. The title defines a displayable label.

```
:: HSActivity s =
 { interaction :: s → Task s
 , procedures  :: [DocumentName]
 , relevance   :: Maybe (s → Bool)
 }
```

The `HSActivity` part of an `HSTask` defines how the task can be completed. For most tasks this is simply a choice between marking the task completed or cancelled. Other tasks embed their own small workflow, such as `outboundCallAction` (explained below). The procedures field allows the specification of a set of procedure documents that contain instructions about the task in natural language. This documentation is made available for quick reference simultaneously with the interaction. The relevance field allows specification of a predicate that tests whether the task is still relevant, given the current information. This is useful when the purpose of the task is to find out some information, but that information has already become available through other means. Based on this predicate, a warning message can be displayed when the task is no longer relevant.

```
:: HSRefinement s =
 { suggested   :: Maybe (s → [HSTask s])
 , alternative :: Maybe (s → [HSTask s])
 , custom      :: s → Task (HSTask s, HSTaskWhen)
 }
```

The `HSTasks` are called hierarchical because each task can be refined into a number of sub-tasks. Broadly defined tasks, like *"Collect as much information about the vessel as possible, from any source"*, need to be further refined during the incident. The `HSRefinement` part of an `HSTask` enables the specification of such refinement. The `suggested` and `alternative` fields contain (optional) functions that compute a set of predefined suggested and alternative `HSTasks` to choose from. Suppose for example that an INMARSAT number of the distressed vessel is known, then a suggested refinement could be *"Search vessel info in INMARSAT database"* while an alternative refinement could be *"Search vessel info with Google"*. To enable improvisation, the custom field specifies the workflow for creating custom refinements. The result of this task is a pair consisting of the refinement and an indication of when the task is to be executed. The `HSTaskWhen` type has the following values:

```
:: HSTaskWhen = HSAlreadyDone DateTime | HSNow | HSAfterTime Time | HSAtTime DateTime
```

Because we observed that urgent actions are often taken first and administrated later, the `HSAlreadyDone` task may be used to add actions that have already been completed.

In the specification, `HSTask` values are never constructed directly, but always via wrapper functions. This makes the definition of concrete actions as concise as possible. For example, consider the following definition for informing a medic at military airfield "De Kooy"[1]:

```
informMedicMVKK :: HSTask Incident
informMedicMVKK = outboundCallAction "informMedicMVKK" "Medical service"
                    "Inform medic on duty at MVKK" (Just (QueryPhonebook "MVKK"))
```

The `outboundCallAction` task defines the workflow for all outgoing telephone calls. Although many telephone calls are made during an incident, it is possible to define a generic workflow that applies to all calls. This ensures that all calls are logged, and that all parties involved in the incident are tracked. To illustrate how this is specified in the iTask WDL we include its definition below:

```
outboundCallAction :: String String String (Maybe ContactHint) → HSTask Incident
outboundCallAction name title description mbHint
  = { meta = meta name title description, activity = customActivity interaction
    , refinement = basicRefinements }
where
 interaction incident
  = defineCaller mbHint incident >>= makeCall incident

 makeCall incident caller
  = connectCall caller >>= select (gotAnswer incident caller) (gotNoAnswer incident caller)
```

[1] Translated only in the paper. In the specification, we use the Dutch descriptions from the procedure documents.

```
gotAnswer incident caller
 =   addLogMessage ("Called with " + visualize caller) incident
 >>= linkContactToIncident caller
 >>| requestConfirmation "Add information"
       "Do you want to add new information to the incident?"
 >>= conditional editIncidentInformation incident

gotNoAnswer incident caller
 =   addLogMessage ("Tried to call " + visualize caller) incident
 >>| requestConfirmation "Try again" "Do you want to try again?"
 >>= conditional (rescheduleCall caller) incident
```

The above example shows that detailed concrete tasks can be specified with a custom `HSActivity` definition. At the same time, we can define flexible, less-detailed tasks. This is illustrated by the `deployOFFSARHeli` task definition below:

```
deployOFFSARHeli :: HSTask Incident
deployOFFSARHeli = multiProceduralAction
  "deployOFFSARHeli" "Deploy OFFSAR Heli" "Deployment of OFFSAR helicopter"
    ["deploy-units.txt","deploy-offsar.txt"] suggested noAlternative
where
  suggested incident = filter (completed incident)
   [ informDCOFFSAR, informHangarOFFSAR, alertCrewOFFSAR ,requestBackupMedic
   , informMedicMVKK, informOperationsMVKK ,requestNOTAM, sendNOTAMRequest
   , requestCHCNetherlands ,informRACAlkmaar, informCSUSecurity, informKMAR, revokeNOTAM ]
```

For this task, it is up to the duty officer to decide which of the suggested refinement actions are actually taken. In this case, no alternative actions are defined, but the suggested tasks and available procedure documentation provide support, while retaining flexibility.

Limitations of the specification

Not all aspects of the Coast Guard's SAR operations are included in the iTask specification. The following limitations can be identified from the SAR application:

- Resource allocation, i.e., the assignment of tasks to workers, is not specified. The workflow is specified without explicit task assignments because the duty and watch officer work as a team. Although they have different roles, each officer can pick up a task for an incident. Hence, work is divided on an ad-hoc basis. Assignment of tasks to teams instead of individuals is not supported in the iTask WDL.
- Planning and scheduling information for future tasks is only implicitly specified. In particular, the iTask WDL only allows the specification of deadlines for tasks, not specific start times.
- The information management task is defined minimally. To make the viewing and editing of all incident-related information manageable, the iTask system should be integrated with a full-fledged information system.
- The specification only contains tasks mentioned in the documents we reviewed. From our interviews and observations, we suspect that there is more structure than is currently documented.

DISCUSSION

Properties of the Specification

With the results of the case study to hand, the first thing we should do is to reflect on what exactly has been specified. This is expressed in the first research question: What are the properties of the specification?

How is it structured and why?

The first thing that is interesting about the structure of the specification is that it is not a simple linear workflow with a clear beginning and end. Rather, it is a combination of two shorter workflows for dealing with new information and for coordinating the incident. The coordination workflow has no pre-defined end, because operations can be aborted at any time. It defines an ongoing (re)assessment of available information and actions that follow from that.

In this respect it is similar to the OODA (Boyd, 1996) view of Command and Control operations. However, we do not specify the task as an explicit loop; in effect, Observe, Orient, Decide, and Act all happen concurrently.

The second interesting aspect is that action coordination is driven completely by human initiative. Actions are suggested but never started automatically. Specifying the workflow as a dependent set of suggestion functions is required to capture the need to adjust the workflow to the specific situation during the incident. Due to the loose organization of tasks, it is impossible to pass information from one task to another, as it is never certain whether a specific task will be executed. Therefore, these tasks have to retrieve the data they need from and store the data they produce into a centrally shared container.

A final point is that, although the workflow to coordinate an incident is organized purely by task suggestions, the workflow of tasks at a more detailed level can be specified as a straightforward sequential/parallel composition of tasks. This allows for a mix of detailed and loosely defined tasks which never overspecify the task, while minimizing underspecification.

Which features of the iTasks WDL are used?

We observe that all iTask core combinators are used to define the inbound communication workflows, the incident response actions, and the generic behaviour of HSTasks. From the basic task primitives, tasks are used mostly for user interaction and data storage. Higher order tasks are used to specify ad-hoc tasks during an incident. The ability of iTask to treat tasks as data and wrap them together with meta-data in single values is used in selecting the suggested and alternative tasks.

Expressiveness of the iTask WDL

The Coast Guard workflow specification reveals intrinsic properties of the iTask WDL. Those aspects that could not be specified show its boundaries, while the use of specific features justifies their inclusion in the language. These topics are covered by the second research question: Is the iTask WDL expressive enough for this case?

Are there aspects that could not be specified using the WDL's primitives?

There are a few aspects of the SAR workflow that could not be specified with the WDL. Resource allocation is not modelled because the WDL only supports the assignment of tasks to individual users, not to multiple users at once. Because the specific assignment of tasks to individuals does not reflect the work on tasks as a team, resource allocation has been omitted from the specification altogether. Planning of future tasks has been included in the specification only implicitly in the form of a line of text added to a task description stating at what time a task should be done. The iTask WDL is only capable of specifying deadlines on tasks, not start times. Hence, the support system generated from the specification does not have the possibility of drawing the user's attention to the task at the scheduled start time.

Another aspect that cannot be defined using the WDL is the structure of the database underlying the `manageInformation` tasks. Technically, this is outside the scope of workflow definition. However, because shared information plays a big role in this case, it would be desirable to have database integration that goes beyond explicit query and update steps in the workflow.

If so, is there a fundamental reason why not? Or can the WDL be improved to include these?

No fundamental issue would preclude extending the iTask WDL to remove the first pair of limitations, i.e., task assignment to teams and scheduled start times. The only reason why these aspects have not been defined is the immaturity of the language. As the language evolves, we expect that each new case study will require some API extension. By contrast, database integration is a fundamental issue. This would make the language a hybrid workflow and information modelling language. However, methods exist for mapping transparently between Clean and databases (Lijnse and Plasmeijer, 2009) which could underlie such a hybrid language.

Are there aspects that could only be specified using the WDL's dynamic features?

It is easy to focus on the limitations of the language because they are clearly revealed by the case study. But, it is just as interesting to reflect on aspects that could be expressed. What this case study shows is that not all tasks can be known in advance, and that the order in which tasks are executed is not fixed. This can only be expressed by a language that determines tasks during execution. A formalism that is only capable of choice between fixed tasks quickly becomes unmanageable under these conditions as n tasks can be executed in $n!$ orderings. Another property

of the work of the Coast Guard is that ad-hoc activities that are not defined in a procedure, but based on experience, common sense, and creative problem solving, are a normal part of the work. It follows that coming up with new tasks is part of the normal workflow. Higher-order task definition are therefore a necessary language feature.

CONCLUSIONS & FUTURE WORK

In this paper, we have explored the use of iTasks to capture crisis response workflow by means of a case study based on the Netherlands Coast Guard's Search And Rescue operations. From this case, we found that, to attain the required flexibility, the workflow needs to be specified as a collection of suggested and alternative actions based on current information, rather than as a statically pre-determined flow. The workflow is highly parallel, with most tasks depending on or contributing to a shared operational picture. Furthermore, we found that improvisation and performing ad-hoc actions are an essential part of the regular SAR workflow.

We concluded that the iTask WDL is expressive enough to capture the loose and parallel structure of tasks and, by using higher-order tasks, the definition of ad-hoc actions. We have captured this structure in a generic hierarchical model, which is reusable for other coordination tasks with improvisation aspects. Nonetheless, the emphasis on a shared operational picture around which the coordination of SAR revolves reveals an opportunity for future work on extending the iTask WDL. The current version of the WDL is designed to specify tasks and the data they use, not to specify information systems. We believe it would be possible to integrate the workflow definition language with a database modelling language using methods from (Lijnse and Plasmeijer, 2009). This would enable a complete executable SAR support system to be specified using a single specification language.

ACKNOWLEDGEMENTS

The authors would like to thank the Netherlands Coast Guard for allowing us to study their operations and so openly providing us with the necessary information and documents, and the anonymous referees for their valuable suggestions.

REFERENCES

1. Boyd, J.R. (1996) The essence of winning and losing, *Unpublished lecture notes*, Maxwell Air Force Base, AL.
2. Fahland, D and Woith, H. (2008) Towards process models for disaster response, *Proceedings of the First International Workshop on Process Management for Highly Dynamic and Pervasive Scenarios.*
3. Jansen J., Lijnse, B. and Plasmeijer, R. (2010) Towards dynamic workflows for crisis management, *Proceedings of the 7th International ISCRAM*, Seattle, WA, USA.
4. Jansen, J., Plasmeijer R., Koopman, P. and Achten, P. (2010) Embedding a web-based workflow management system in a functional language *Proceedings 10th Workshop on Language Descriptions, Tools and Applications, LDTA '10,* pages 79–93, Paphos, Cyprus.
5. Lijnse, B. and Plasmeijer, R. (2009) Between types and tables - Using generic programming for automated mapping between data types and relational databases, *Revised Selected Papers of the 20th International Symposium on the Implementation and Application of Functional Languages, IFL '08*, To appear in Springer LNCS 5836.
6. Peukert, H., Lincourt, D. and Zimmermann, B. (2009) Support for agile planning & execution of coordinated actions, *Proceedings of 14th ICCRTS C2 and Agility*.
7. Plasmeijer, R., Achten, P. and Koopman, P. (2007) iTasks: executable specifications of interactive work flow systems for the web, *Proceedings of the International Conference on Functional Programming, ICFP '07*, pages 141–152, Freiburg, Germany. ACM Press.
8. Sell, C. and Braun, I. (2009) Using a workflow management system to manage emergency plans. *Proceedings of the 6th International ISCRAM Conference* - Gothenburg, Sweden.