15th ICCRTS

'The Evolution of C2'

Web Based Dynamic Workflow Systems for C2 of Military Operations

Topics: (3) Information Sharing and Collaboration Processes and Behaviors,
(9) C2 Architectures and Technologies,

Jan Martin Jansen, Tim Grant (NLDA),
Bas Lijnse (NLDA & Radboud University Nijmegen, the Netherlands),
Rinus Plasmeijer (Radboud University Nijmegen, the Netherlands)

POC: Jan Martin Jansen

Netherlands Defence Academy (NLDA)
P.O. Box 10000, 1780 CA Den Helder, the Netherlands
Tel: +31 223 657127
jm.jansen.04@nlda.nl

Web Based Dynamic Workflow Systems for C2 of Military Operations

Abstract Modern military operations are complex endeavours involving different services and nations, as well as governmental, commercial, non-governmental, and international organizations. Each partner may have its own planning process and tools. This diversity must be orchestrated to plan the overall operation, while maintaining the agility to respond to changing situations.

We contend that dynamic workflow mechanisms are suited to planning military operations. We developed the iTask dynamic workflow system that enables the construction of high level multi-user workflow applications. Workflows can change in response to dynamic situations, new tasks can be spawned by or be dependent on previous tasks, tasks can be dynamically adapted. iTask based applications have a web-based user interface, allowing external partners to use them without installing special software. Moreover, new parties can join them on-the-fly. Because of its focus on dynamic processes iTask appears promising for development of flexible C2 systems.

In this paper we present a discussion of the potential of the iTask system for building C2 systems. We give an overview of the system, and discuss to what extent it meets the requirements of the C2 domain. We also sketch a number of promising application areas in this domain.

1 Introduction

In modern warfare many activities have to be deployed by many people using a great diversity of systems. The coordination and control of these activities is becoming more and more complex. The advent of NCW and NEC [AGS99] complicates this even more, because much more information is becoming available in an even shorter time frame. Decisions are not taken in centralized headquarters anymore, but are the result of a collaborative effort of many. Command and Control has extended from a single system/group activity to a networked activity involving many systems and people distributed over large areas. Often non-military like local authorities and non governmental organizations (NGO's) are involved in operations. Also the nature of military operations has changed. Asymmetrical operations have become the standard. Information is the most important weapon in these operations. But obtaining information is difficult and requires other sources then the traditional sensors. Instead complex intelligence operations are required. Through these developments the borderline between military operations and response operations for crises, whether caused by aggression or caused by accidents or natural disasters, is fading. Systems that take care that information is made available to the right persons at the right moment, and that support the gathering of information become ever more crucial for successful execution of operations.

In recent years, the focus for research has been on the development of systems to enhance the quick sharing of information using Web 2.0 technology and Service Oriented Architectures (SOA, [IEH09]). Although making information available to all parties is crucial to enhance situation awareness [EBJ06], a recurrent theme in our discussions with military and crisis management professionals has been that the real challenge is coordination and control. At first glance, Workflow Management Systems appear to have potential to support this (see also [SB09], [PLZ09] and [FW08]). WFMS's are computer applications that coordinate, generate, and monitor tasks to be performed by human workers and computers. Every activity in an 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 [SB09,PLZ09,FW08]. First, contemporary workflow systems are commonly rather rigid because they only model the static flow of control. Second, in many cases the activities to be conducted for executing an operation 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. They only offer the execution of detailed predefined plans. In other words, contemporary workflow systems are not capable of dealing with the dynamic nature of modern military and crisis response operations where tasks may heavily depend on the outcome of previous tasks and plans must be changed on-the-fly due to changing circumstances.

Recent work on the use of functional programming techniques for workflow modeling has led to the development of the iTask system [PAK07,PAK08a]. 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 allows for on-the-fly (dynamic) adaptation of tasks. A full (extended prototype) implementation of the iTask system is

already available. A number of iTask applications have been implemented including a conference management system (see [PAK+08b]) and a number of smaller example applications. It should be noted that iTask is not a C2 system itself, but a toolkit for the construction of such systems.

In this paper, we present a discussion on the suitability of dynamic workflow specification, and its implementation using the iTask system, for modern C2 of both military and crisis response operations. We also sketch a number of candidate application areas for iTask and indicate what the possible gains are for these areas. We use five key design requirements for response technology proposed by Jul in [Jul07] as a framework for structuring our discussion. The current iTask system already offers important functionality for supporting C2 operations, and can be trivially extended to meet even more. However, we also identified a number of research challenges that need to be addressed to fully optimize the iTask system for supporting C2. Although some of the strengths, weaknesses and challenges we discuss apply only to the iTask system, most apply to workflow management systems in general.

2 The iTask system

The iTask system (itasks.cs.ru.nl) is a domain specific workflow language embedded in the functional programming language Clean, enabling the creation of dynamic data dependent workflow systems. This means that it enables programming of workflow systems in a programming language that is specifically tailored for this purpose, but at the same time has the full computational expressiveness of a modern functional language. A workflow system is data dependent if it allows for adaptation of workflows using intermediate results. 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 workflow applications are generated that run on the web, optionally distributed over servers and clients [PJKA08]. The iTask system is based on open web-standards and can therefore be accessed by anyone who has access to Internet, nowadays including many mobile devices. The system has an interface resembling e-mail client software to reduce the need for users to learn additional interactions.

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*. Result types are not limited to simple data such as integers, records, etc., but can also be documents, or even new 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. This also holds for tasks that produce or consume other tasks.

We distinguish two kinds of tasks: basic tasks and composed tasks. Basic tasks are elementary tasks that can be fulfilled by one user in one step. In the workflow language these are black box primitives that are implemented at a lower level. An example of a basic task would be the entering of information in a web-form by a user. The result of this task is then the entered data. Composed tasks, or workflows, are defined by composition of (basic) tasks using so-called *task combinators*.

2.1 Basic iTasks

The iTask standard library offers several functions for creating basic units of work: basic tasks. An important example is the *generic* task where a user is asked to supply information. The generation of a web-form to enter this information and the processing of its result are handled fully automatically by the iTask system. In this way one can create data entry tasks in just a single line of code. Figure 1 shows the code for a task where the user can supply the information for a military mission, together with a picture of the generated form. The code consists of data type definitions and a task definition (`enterMission`) using these data types. This example also shows that documents can be attached to tasks.

```
:: Mission =
    { type        :: MissionType
    , date        :: Date
    , time        :: Time
    , nrTroops    :: Int
    , location    :: Location
    , moreDetails :: Bool
    , description :: Document
    }

:: MissionType = PeaceKeeping | CounterTerrorism |
                 SpecialService | IntelOperation |
                 War | Other String

::Location = {city::String,country::String}

enterMission :: Task Mission
enterMission = enterInformation "Please provide information about the mission"
```

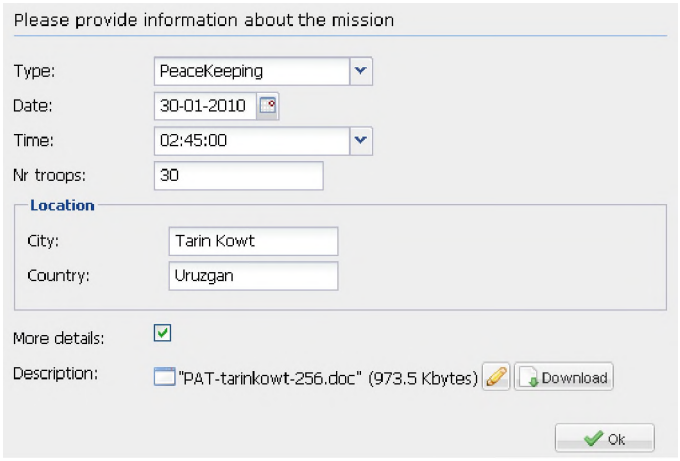

Fig. 1. A Generic Data Entry Task for `Mission` Data.

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 fallback based on manul 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: requesting lists of 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, databases, sensors, or web services (for the exchange of information). We have for example implemented access to Google Maps.

2.2 iTask Combinators

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.

Tasks can be organized in many ways. In most contemporary workflow systems organization is limited to a fixed set of patterns (see [AHKB02]). Because the iTask language is embedded in an expressive general purpose host language (Clean) all common patterns, and many more can be expressed using using relatively few combinators. Here we discuss the most important ones.

Sequential Combination In contrast to most workflow specification languages, information is passed explicitly from one task to another in the iTask formalism. 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
```

Note that 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 (mostly graphical) 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.

Parallel Combination An important combinator for executing a number of tasks in parallel is the `parallel` combinator. Where other workflow formalisms contain a large number of patterns (see [AHKB02]) for executing tasks in parallel, iTask needs only one combinator for this. Using the power of the functional host language, one can construct all other patterns (and more) using this single combinator. This is hard to do in other workflow languages because these lack the right abstraction mechanism for realizing this. With the parallel combinator one can start the execution of several tasks in parallel and stop this execution as soon as a user specified condition is fulfilled. For example, one can stop when one task (or-parallelism) is finished:

```
anyTask [task_1,task_2,task_3,task_n]
```

When all tasks (and-parallelism) are finished:

```
allTasks [task_1,task_2,task_3,task_n]
```

Or when the results of the finished tasks satisfy a certain condition (ad-hoc parallelism):

```
conditionTasks condition [task_1,task_2,task_3,task_n]
```

These different combinators are all shorthands for the same generic `parallel` combinator instantiated with different parameters.

Task Assignment Tasks can be explicitly assigned to users using the task assignment (@:) combinator.

```
userid @: task
```

Here the task `task` is assigned to the user with login name `userid`. The user to whom a task is assigned, can be entered explicitly in the workflow model. Alternatively, it is possible that during the execution of the workflow, a task determines the user to which another task must be assigned. Once tasks have been initially assigned, it is always possible to reassign them to another user on-the-fly. It is possible to monitor the progress of tasks. This information can be used to re-allocate task to different users, to stop task or to replace tasks by other tasks.

2.3 An Example iTask Workflow

A typical example of a task for which a dynamic workflow system can be used is the dynamic allocation of resources. Consider, for example, the following scenario. For a complex military mission transportation of people, equipment and supplies is necessary. The amount and kind of transportation devices heavily depends on the location of the mission area, the number of people and goods to be transported, the condition and safety of the transportation routes.

We implemented a prototype application that automates this process using the iTask system.

The starting point for this workflow is a mission report like the one described above. This report contains the type and the location of the mission. The workflow uses this information and a set of available transportation providers and their locations to calculate an initial set of requests to be sent to transportation providers to obtain the right amount of vehicles. Each provider should reply within a certain time limit whether it is capable of supplying the requested amount. In case not enough vehicles can be supplied the system automatically starts requesting other providers and recursively continues doing this until enough transportation capacity is available.

Due to space limitations, we only show the code of the first part of the workflow specification. This part handles an incoming report for a mission and starts the operation. It consists of three steps: First some data describing the mission is entered (`enterMission`), then the appropriate action are chosen (`planActions`, and finally the actions set in motion (`allTasks`). During the second step, a suggestion for further action is computed based on the type of mission that is entered in the first step. During the third step all tasks that have been chosen as action are carried out in parallel. Transportation is handled in the `orderTransport` task. The iTask code for this workflow is the following:

```
startMission
  = enterMission >>= planActions >>= allTasks
where
  enterMission :: Task Mission
  enterMission = enterInformation "Please provide information about the mission"

  planActions :: Mission → Task [Task Void]
  planActions mission
    = updateMultipleChoice "Choose actions" options (suggestion mission.type) ++
      if mission.moreDetails [detailSpecification] []
```

```
where
  //Generate the list of possible tasks to choose from
  options = [f mission \\ f ← [makeComsPlan,orderTransport,orderSupplies,orderAirSupport]]

  //Compute the indexes in the options list that are initially selected
  suggestion PeaceKeeping      = [0,1,2]
  suggestion CounterTerrorism  = [0,1,2,3]
  suggestion SpecialService    = [0,1]
  suggestion IntelOperation    = [0]
  .........
  suggestion _                 = []
```

This small piece of code already demonstrates two core features of the language. First, it integrates computation in the workflow. The `suggestion` function computes the initial selection of actions from the information that is entered in the `enterMission` step. In this case it is a simple one-to-one mapping of mission kinds to selections, but it is possible to do any computation to select or parameterize the next steps in a workflow. The second interesting feature is that the result type of the `planActions` task is a list of tasks. This list of tasks, which are all parameterized with the mission information, is then executed in parallel by the `allTasks` combinator. The possibility to have tasks that have new tasks as their result can be used to create highly dynamic models that contain steps in which parts of the worklow are interactively defined during execution. In `planActions` we also inspect the `moreDetails` field in the original form to decide whether or not a `detailSpecification` task should be started parallel to the other tasks.

2.4 Dynamic Behavior: Exceptions and Change

Several authors, like [SB09], [PLZ09] and [FW08], have already indicated that workflows need to be adaptive to be of use for complex domains like command and control and crisis management operations. The iTask system offers a number of programming constructs to support the following kinds of dynamic behavior:

1. Dynamic behavior that can be anticipated and where the normal course of actions is not affected. In these cases the procedure to be followed depends on the intermediate results of previous tasks. This is considered as normal dynamic behavior and is provided by the sequence (>>=) combinator.
2. Dynamic behavior that can be anticipated where the normal course of actions is affected. In these cases normal procedures should be stopped and a different procedure should be started. The *exception* mechanism in the iTask system provides this capability.
3. Dynamic behavior that cannot be anticipated and detected within the workflow itself. In this case ad-hoc changes should be made to one or several workflows. The normal procedure should be stopped, and then either a different procedure should be started or an adaptation to a (sub)task should be made. This form of dynamics is provided by the *change* concept in iTask

iTask supports an exception mechanism similar to what is found in common programming languages. 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. This handler can now start a new task using information raised in the exception as its input. An exception must be explicitly thrown in a workflow. So, the designer of the workflow must be aware that exceptional situations may occur during the execution of a workflow and therefore has to define a handler for them. Exceptions enable the separation of uncommon borderline cases from the regular workflow.

The *change* concept is complementary to that of the exception. While an exception is the result of an abnormality that occurs within a workflow, a change is triggered from outside the specified workflow. Tasks on which people are working can be replaced on-the-fly with other tasks. Because iTask workflows are typed, the new task should return a result of the same type as the replaced task. An example of a change is the replacement of a complex process by an ad-hoc made to-do list, in case the user has determined that the process is inappropriate for the current situation. Another example is the replacement of a process by the ad-hoc entering of a result that is obtained in another way (not using the workflow system).

2.5 The iTask Client

In the iTask system, workflow instances are executed by a server application that is generated from the workflow specifications. These server applications dissiminate information about tasks through a collection of web services.

Because end-users cannot access such services directly, the iTask toolkit provides a generic client application to let people view, and work on tasks. This client application, shown in Figure 2, is an Ajax application which is similar to a web-based e-mail client. But instead of an inbox of messages there is an inbox with interactive tasks. Because web-based e-mail applications are very common, this design requires a minimal amount of addtional learning.

The names of the tasks that the worker needs to perform are presented in the *task list* displayed in the upper right pane. This pane can be compared with the list of incoming e-mails. When the worker clicks on a task in the task list, its current state is displayed in the lower right *task pane.* Tasks can be selected from the task list in any order, allowing a local operator to determine a preferred order of execution. The iTask toolkit automatically keeps track of all progress, even if the user quits the system. When a task is finished, it is removed from the task list. Workers can start new workflows, by selecting them in the left *workflow pane.* In general any number of workflows can be started. The task list is updated when new tasks are generated, either on the users own initiative, or because they have been delegated to it. The entire interface is generated automatically from a workflow specification.

3 Example Applications in the Military Domain

The introduction to the iTask system in the previous section, is necessarily dense and abstract due to the meta-system nature of the toolkit. Therefore, before continuing with a discussion of its suitability for military / crisis response operations based on requirements from literature, we first

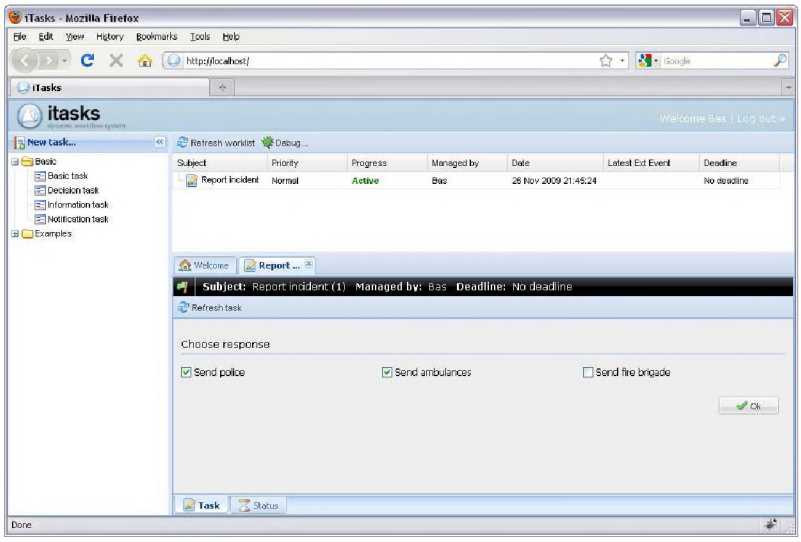

Fig. 2. A Screenshot of the iTask client application

give some envisioned example applications areas in this section. Contrary to the iTask system itself, which is an implemented proof-of-concept system (previously published in [PAK07,PAK08a]), these applications are just examples to sketch a more concrete vision of its use in a military context.

3.1 Useful Characteristics

The iTask system can be used to make applications that tell people what to do at what moment. In this way iTask does not differ from other workflow systems and planning tools. More interesting are the unique properties of iTask that cannot be found in other workflow management systems.

First, workflows application programmed in iTask's are flexible in many ways. Because iTask allows for data dependent workflows intermediate results can be used to parametrize future steps. For example, if for transportation more vehicles are needed than can be supplied by the standard transport service, an additional workflow can be started for obtaining more transport capacity. Using iTask's dynamic data dependencies many dynamic aspects of operations can be captured. But not all operations can be captured in predefined plans, because the steps needed to be taken during the operation cannot be determined exactly beforehand. But even in those cases the generic structure of the process is mostly known and only in the more detailed sub-procedures ad-hoc actions involving human improvisation are needed. For this we can use the *Change* concept of iTask. Using this *Change* concept workflows can be adapted in many ways. The most direct way to use it, is the replacement of (part of) a workflow by ad-hoc entering of information. This is necessary when someone decides that this part of the workflow is not appropriate for the current situation. Instead, the information needed at this point of the workflow is obtained in another (ad-hoc) way. This seems to be a trivial issue, but one often has 'to fight the system' because a procedure is not appropriate for the current situation, but one has to continue because there is no way to circumvent the system. The second way to use the *Change* concept is to do the exact opposite. In this case it is

not possible at design time to give an appropriate workflow for a subtask at a certain point in the workflow definition. In this case a default workflow that just consists of a form where the user has to enter the appropriate information is offered in the workflow definition. During the execution of the workflow the user can decide to start a dedicated workflow that supplies the information needed at that point. The information that is generated by this workflow is now automatically entered into the other workflow.

Second, the exception mechanism can be used to stop an already running workflow automatically and replace it with another workflow. For example, a military patrol must be aborted, because new information shows that it is too dangerous to continue. Instead, an air strike action is necessary to clear the area. Stopping a workflow can also be done by a user with the appropriate rights. In case (part of) a workflow is stopped, the users involved in it are automatically notified and the results already obtained are discarded.

Third, it is possible for a user to construct ad-hoc workflows by making a composition of already existing workflows. The user is offered a dedicated interface to make simple sequential and parallel compositions of existing workflows.

Fourth, iTask is not a closed system and allows for easy integration of other web-services. This offers a straightforward way to extend the functionality of an iTask application and to use iTask as a web services coordination tool.

3.2 Preparation of Deployment for Military and Peace Keeping Operations

Preparations for military operations like the deployment of troops for peace keeping operations are characterized by their complexity and unpredictability. They are complex because large amounts of people and equipment have to be transported to very remote locations and every deployment has its own unique characteristics. They are unpredictable because it is almost impossible to use ready made plans to execute them and existing plans often have to be adapted due to unforeseen circumstances. The planning and execution of deployments comprises the following aspects:

Logistics Before people can be deployed, accommodation, power, water and food supply, etc. have to be arranged.

Transport Transportation is needed both for people and material (accommodation and supplies). A large part of the transportation has to be done beforehand (accommodation, infrastructure). Other transportation is needed during the entire deployment (food, fuel, ammunition, replacements).

Intel Prior to the deployment, but also during the operation, intelligence operations are needed. Examples of prior Intel requirements are: What are the expected enemy forces, what is the available infra structure (communication, resources (water, food etc))? What are safe routes for transportation? What are the local terrain conditions? How is the local climate? What kind of protection is needed for the initial transports? Examples of Intel during the deployment are: What is the enemy behavior? What is the attitude of the local civilians?

C2 & Communication A command and control and a communication infrastructure has to be built-up for the operation: radio, telephone (including GSM), satellite for communication with

headquarters and allies including Non Governmental Organisations (NGOs), computer networks for the exchange of information, encryption equipment, Internet for welfare communication.

Procurement Often special equipment and goods have to be procured for the mission. Standard ordering procedures often have to be circumvented because they take too long and ad-hoc procedures have to be used instead.

Protection What kind of weapon and sensor systems have to be used? Are they allowed by the Rules of Engagement (ROE)?

Budget What will be the costs of the deployment? How do we stay within the maximum allowed budget limits?

These processes can be very dynamic, because, for example, Intel information, changing ROE's and political involvement can influence already started tasks. Planning of these complex operations often involves the commitment of large numbers of geographical distributed staff personnel over periods varying from several weeks to several months. Currently, normal communication channels like telephone and e-mail are used for the exchange of information, while in general spreadsheet and database applications are used for maintaining information, mostly on an individual or small departmental basis. This means that other people and departments do not have insight into this information and should make explicit requests (by telephone or e-mail) to obtain it. Moreover, one has to deal with international partners, both military and civil with which plans have to be aligned.

It is clear that dynamic workflow applications can be of great help for planning these operations. We summarize a number of issues that can be supported:

- support of the overall structure of the entire process;
- supplying the right information to the right parties at the right moment. We are dealing with a variable number of dislocated people causing a dynamic topology;
- the automatic checking of deadlines and taking actions in case they are passed;
- interrupting activities already started due to changed circumstances;
- monitoring the status of actions with the possibility to interrupt or reallocate tasks (automatically or by users);
- the ad-hoc creation or adaptation of workflows for subtasks by users;
- automatically monitoring and checking of budget.

3.3 Intelligence Operations in Asymmetric Warfare

Intelligence operations are becoming a more and more important part of modern warfare. Especially in counter terrorism the timely gathering of information about plans of adversaries is the most important weapon against them. Many people and systems are involved in this gathering of information. As a result a large amount of information is generated, which easily leads to an information overload and, as a result, important information is often not available at the right moment at the right place. Using a dynamic workflow system like can help to structure the information streams in this information gathering. For example, a local agent may obtain information about a possible adversary. A workflow can now be started and as a first step the user has to enter information in a form. This information can be used by the system to start a workflow that takes care

that appropriate actions are taken. Using the automatic monitoring and timeout features of iTask special actions can be taken in case insufficient progress is made.

3.4 Crisis Management and Civil Military cooperation

In crisis management operations one is often confronted with situations where people of many different organizations have to cooperate, often in an ad-hoc manner, to tackle the crisis. An important issue here is that these different organization all may have their own command and control procedures and systems (stove pipes). Dynamic workflow applications can be used to integrate information from different systems, and to supply the overall command and control for operations. For these kinds of collaborations it is important that different organization can easily join the system without the need to install special software.

Crisis Management operations are in general very unpredictable and it is therefore impossible to capture their command and control in a predefined dynamic workflow. But the generic structure of these operations can be captured in a workflow and for many more detailed subtasks dedicated workflows can be defined. Having a tool that allows for using human improvisation to combine a generic structure with detailed workflows for subtasks can be of great help.

4 Strengths and Weaknesses

Military operations and civilian, or joint civil-military crisis response operations share many characteristics. Both have to deal with complex resource allocation and complex information management in a potentially hostile environment. Taking into consideration a further convergence of military and joint civil-military operations we view these domains therefore as single broad domain. This raises the bar somewhat in comparison with pure military C2 systems, because existing structures, such as an established chain of command, or a known level of training cannot be assumed to be available.

Because the iTask workflow language is embedded in a general purpose programming language, it can in principle be used to construct any C2 support, crisis response, or other process support system imaginable. However, the required effort that is needed and the quality of the resulting system is determined largely by what is offered out-of-the-box. Therefore, it is important to know whether what is currently offered by the iTask toolkit matches the needs of the domain for which one aims to build systems for.

When proposing technological solutions, different authors highlight different requirements as being important (see for example [IEH09] and [SB09]). To determine the status quo of the iTask system's applicability for the joint crisis management / military domain and to identify areas for future research without bias, we need an independently defined set of requirements. For the crisis management domain, Jul in [Jul07] provides such a set of five design requirements distilled from an analysis of the domain:

- **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.

In this section we systematically discuss the strengths and weaknesses of the iTask system in view of each of these requirements. The purpose of this discussion is not to evaluate whether the system, in its current form, is ready and useful for deployment during a crisis response or military operation. Its primary goal is to uncover interesting challenges to focus further research.

4.1 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 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. Currently, the iTask system does not yet offer any support for learning at the task level.

4.2 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 at run-time that 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.

4.3 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. When tasks are delegated to others, the user who delegated them can track them. 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 stimulate the involvement of 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 property of the iTask system that influences the possibility of defining "cooperation friendly" workflow specification is the focus on users as individuals. Tasks are always assigned to, and managed by, a single individual. Social 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.

A final issue is the ability to cooperatively define and plan tasks. By default, iTask workflow models are controlling tasks in a top-down manner, assigning tasks to users as planned in a workflow specification. However, because tasks can be results of other tasks and tasks can receive tasks as their input, it is possible to define meta-workflows that let users agree upon a set of tasks and their order to define a new workflow.

4.4 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 three different levels, as explained earlier, it is potentially capable of complete compliance with this fourth requirement.

However, we should not claim victory too soon. Although it is technically possible to define very flexible workflow models, we must acknowledge that the usefulness of this expressive power is constrained by the interface through which it is exposed to end-users. Additional research is needed to develop generically applicable problemsolving 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.

4.5 Requirement #5: Graceful Augmentation

Meeting this final requirement completely is near impossible for any workflow system because removal of a workflow system during the execution of an operation guided by it will cause disruption. 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 (see [PJKA08]), by transferring part of the workflow computation to the client system. It is, of course, possible to use the system in a controlled environment such as a command post while communicating tasks through other channels, where it could still have advantages over written handbooks, because iTask workflow specifications are dynamic.

4.6 Additional opportunities

Because the requirements suggested by Jul cover crisis response technology in a very broad sense, there are properties of the use of dynamic workflow models to support operations that cannot be linked directly to one of the requirements, but are potentially valuable. Examples include:

- **Verification through formalization and prototyping:** In written plans and procedures, anything can be described, even when logically contradictory, ambiguous, or otherwise incorrect. By formalizing workflows in a modeling formalism, one is forced to write down precisely what the steps in the process are. But even then, workflows can be defined that are logically sound, but nonetheless make no sense at all. Because the iTask system can generate executable systems instantly from models, it is possible to rapid prototype workflow models and to verify them through simulation and testing during the design phase.
- **Data to create situation awareness:** When processes and actions are coordinated and communicated through a workflow support system, there is an abundance of data available during operations about what tasks people are working on and what processes are currently running. This data, when presented in the right way to the right people, could be valuable for assessing the situation and for planning further action. Although this data is unavoidably incomplete and it is not immediately clear how to extract useful information from it, it offers interesting opportunities. Further research is needed.
- **Data for evaluation and learning:** The availability of data about tasks and processes is not only useful during operations, but may also be utilized afterwards to evaluate an operation and learn from the mistakes that were made.

5 Future Challenges

From the discussion of the iTask system in the previous section we can conclude that, although there is substantial potential that certainly justifies further research, it is not the perfect programming toolkit for building C2 or crisis management systems yet. There are still challenges that have to be tackled.

5.1 Collaboration

One area where the iTask system could gain greatly is in the facilitation of collaborative work. The current focus on individual users, without the concept of (informal) organizations, limits collaboration or partnership formation. The communication through formal task assignment only also limits its potential.

Quick wins can be achieved by integrating easy-to-build communication features such as chat sessions linked to tasks or enabling users to let others view the tasks they are working on. Although this would make it easier to get help with, or give feedback on tasks, a much bigger challenge lies in the integration of social constructs like organizations, (temporary) teams, partnerships or friends. This would decouple the direct relation between a task and an individual person and raises questions about dealing with concurrency, shared responsibility, shared decision making and individuals performing tasks on behalf of organizations.

Another way of facilitating the creation of cooperation friendly workflow models could be the development of out-of-the box meta-workflows for collaboratively defining and assigning tasks.

5.2 Effective Flexibility

Another challenge for iTask's design would be to apply the power of adapting running processes to resolve unexpected problems that arise during operations. Although it is technically possible to adapt workflows that are already running, two important questions that would have to be answered are: First, how will users know that the workflow they are executing is not going to fulfill its goal? And second, how should they instruct the system to change the workflow to resolve the problem?

To answer the first question, more research is required into what information about a workflow instance is needed by users to be able to detect that there is a problem. A related issue is whether it is possible to monitor progress automatically and to warn users of an unexpected lack of progress.

The second question is possibly even more challenging. An easy way out would be to let end-users solve the problem by providing some (visual) programming interface to specify alternative workflows. However, this assumes that all users can, and want to use this when facing an immediate problem. We believe the bigger challenge is the design of an interface to the underlying workflow model that helps stranded users in either resolving their immediate problems and continue with minimum disruption, or let them gracefully abort.

5.3 Domain Specific Frameworks

The design of workflow specifications is likely to be influenced by which basic tasks, combinators and generic subprocesses are readily available. For example, if there are meta-workflows supporting collaborative task assignment available in a library, it is more likely that collaborative steps will be incorporated in a workflow than when the collaboration process itself also has to be specified. It is therefore necessary to have available a collection of tasks, data types and generic workflows that are common in the domain. A major challenge will be the design of a domain-specific framework that supplements the generic iTask system to create a platform for building workflow support systems to aid crisis management operations.

6 Conclusions

In this paper we presented dynamic workflow programming, as implemented by the iTask system, as a candidate platform for developing applications to support command and control of military and crisis management operations. Because of its unique features like: data driven, parameterizable workflows and extensive support of dynamic behavior we view it as a potentially valuable tool for construction of C2 systems for this domain. We have explained the basics of programming such systems using the iTask toolkit, and sketched our vision of possible applications in the military and crisis management domains, which we consider a single domain in this context. Most notably, we have compared the current iTask system to independently defined requirements for technology during crises defined by Jul in [Jul07]. We have discussed the strengths and weaknesses of the iTask system in light of these requirements to identify future research challenges. Based on this comparison, we are confirmed in view that dynamic workflow programming is indeed potentially valuable, but research challenges are: facilitation of collaboration on tasks, interaction with the workflow model during execution, and the need for domain specific frameworks. By focusing research effort on these issues, we hope to develop the system further, into a valuable C2 construction toolkit for building systems that flexibly support people under demanding circumstances.

References

[AGS99] D.S. Alberts, J.J. Garstka, and F.P. Stein. *Network centric warfare: Developing and leveraging information superiority.* C4ISR Cooperative Research Program Publications Series, Department of Defense, 1999.

[AHKB02] Wil van der Aalst, Arthur ter Hofstede, Bartek Kiepuszewski, and Ana Barros. Workflow patterns. QUT technical report, FIT-TR-2002-02, Queensland University of Technology, Brisbane, Australia, 2002.

[EBJ06] M.R. Endsley, B. Bolte, and D.G. Jones. *Desining for sitation awareness: An approach to user centered design.* Taylor & Francis, London, 2006.

[FW08] Dirk Fahland and Heiko Woith. Towards process models for disaster response. In M. de Leoni, S. Dustdar, and A.t. Hofstede, 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)*, September 2008.

[IEH09] Magnus Ingmarsson, Henrik Eriksson, and Niklas Hallberg. Exploring development of service-oriented c2 systems for emergency response. In J. Landgren and S. Jul, editors, *Proceedings of the 6th International ISCRAM Conference - Gothenburg, Sweden*, May 2009.

[Jul07] Susanne Jul. Who is Really on First? A Domain-Level User, Task and Context Analysis for Response Technology. In (B. Van de Walle, P. Burghardt, and C. Nieuwenhuis, editors, *Proceedings of the 5th International ISCRAM Conference - Delft, the Netherlands*, May 2007.

[PAK07] Rinus Plasmeijer, Peter Achten, and Pieter Koopman. 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, 1-3, October 2007. ACM Press.

[PAK08a] Rinus Plasmeijer, Peter Achten, and Pieter Koopman. An introduction to iTasks: defining interactive work flows for the web. In *Selected Lectures of the 2nd Central European Functional Programming School, CEFP'07*, volume 5161 of *LNCS*, pages 1–40, Cluj-Napoca, Romania, 2008. Springer-Verlag.

[PAK+08b] Rinus Plasmeijer, Peter Achten, Pieter Koopman, Bas Lijnse, and Thomas van Noort. An iTask case study: a conference management system. In *Selected Lectures of the 6th International Summer School on Advanced Functional Programming, AFP'08*, volume 5832 of *LNCS*, Center Parcs "Het Heijderbos", The Netherlands, 19-24, May 2008. Springer-Verlag.

[PJKA08] Rinus Plasmeijer, Jan Martin Jansen, Pieter Koopman, and Peter Achten. 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, 15-17, July 2008.

[PLZ09] Hans Peukert, David Lincourt, and Birgit Zimmermann. Support for agile planning & execution of coordinated actions. In *Proceedings of 14th ICCRTS C2 and Agility*, 2009.

[SB09] Christian Sell and Iris Braun. Using a workflows management system to manage emergency plans. In J. Landgren and S. Jul, editors, *Proceedings of the 6th International ISCRAM Conference - Gothenburg, Sweden*, May 2009.