Information Supply Mechanisms in Ubiquitous Computing, Crisis Management and Workflow Modelling

Jurriaan van Diggelen, Robbert-Jan Beun, Rogier M. van Eijk,
Peter J. Werkhoven

Institute of Information and Computing Sciences
Utrecht University, the Netherlands
{jurriaan,rj,rogier}@cs.uu.nl; peter.werkhoven@tno.nl

Abstract. The successful application of ubiquitous computing in crisis management requires a thorough understanding of the mechanisms that extract information from sensors and communicate it via PDA's to crisis workers. Whereas query and subscribe protocols are well studied mechanisms for information exchange between different computers, it is not straightforward how to apply them for communication between a computer and a human crisis worker, with limited cognitive resources. To examine the imposed cognitive load, we focus on the relation of the information supply mechanism with the workflow, or task model, of the crisis worker. We formalize workflows and interaction mechanisms in colored Petri nets, specify various ways to relate them and discuss their pros and cons.

Keywords: Ubiquitous Computing, Notification Systems, Human-machine Interaction, Workflow Modelling, Petri Nets

1 Introduction

Ubiquitous computing [20] is a model of human-computer interaction which offers specific application possibilities and which requires specific design methodologies. In this paper, we will study the use of Workflow Modelling (WM) for designing Ubiquitous Computing (UC) systems, in the domain of Crisis Management (CM). We shall briefly explain these three disciplines and their relations below.

CM involves identifying an incident or a disaster, such as fire or a traffic accident, and subsequently confronting and resolving it in order to minimize the damage. Information transfer plays a crucial role in these activities. Lack of information (*information underload* [11]) is often identified as a potential cause of mistakes as it leads to decisions based on incomplete information. Also, too much information (*cognitive overload* [10]) may cause errors, as it distracts the crisis worker from his or her primary tasks.

Ubiquitous computing provides an adequate way to bridge the information gap in CM. UC aims at making hundreds of networked computing devices and sensors work together to get the *right* information to the *right* person at the *right* time [4]. What qualifies as *right* information then depends on the work that is performed by the crisis team member.

This is how Workflow Modelling fits in. WM has proven itself as a successful method to precisely describe a business process and optimize various aspects such as efficiency, average completion time, and utilization of resources. In our opinion, WM is also a promising approach to model work processes in the CM domain. In general, we believe that WM yields valuable insights in the design phase of *any* UC system. The reason for this is simple. If we expect a system to pro-actively present valuable information to its user (as in UC), the system must know something about the user's *task*. In this paper, we will use the term workflow interchangeably with the term task, although in the literature the two terms are sometimes used to denote slightly different things (we will come back to this issue in Section 5). Therefore, WM can be regarded as a necessary part of the design phase of the system.

Whereas UC, CM and WM are all well-developed research areas, the combination of the three disciplines raises several issues that have not yet been addressed in the literature. In this paper, we tackle two of these issues, which are briefly described below.

The first issue concerns the application of WM to CM. Current workflow management techniques are typically tailored to business processes. Likewise, current workflow analysis techniques are typically concerned with business goals, e.g. minimizing production costs. To apply WM to the CM domain, the important aspects of CM should be well-representable, such as ignorance, information underload, cognitive overload and distraction. In this paper we will apply a WM technique which uses Petri Nets [16]. We will show how the knowledge of the crisis worker can be modelled in this approach. Furthermore, we will show how notions such as information underload and cognitive overload can be mapped to well-known theoretical properties of Petri Nets.

The second issue concerns the application of WM to UC, i.e. the modelling of the interaction mechanisms that are responsible for providing the crisis worker with the right information. Two well-known interaction mechanisms in UC (and multi-agent systems in general) are the Query protocol and the Publish/Subscribe protocol [3]. Both of these protocols have been specified using various formal methods, among which Petri Nets. Nevertheless, these specifications focus on the low level properties of the interaction mechanisms, such as possible network failures, input buffer overloads and so forth. Because UC and CM require us to consider the cognitive aspects of information exchange, these specifications do not suffice. We will show that, by modelling various interaction mechanisms directly in the workflow model, we can examine the effects of these interaction mechanisms in terms of cognitive aspects. In fact, it appears that, next to the query protocol, at least four different types of subscription mechanisms exist. For each of the interaction mechanisms, we will give a Petri net specification and give template design procedures to join these with the workflow model. Furthermore, we will compare the different interaction mechanisms by evaluating their effects on resolving information underload and preventing cognitive overload.

Section 2 presents workflow modelling. Section 3 discusses how information supply mechanisms can be formalized in relation to a workflow. Analysis techniques are discussed in Section 4. Related work is discussed in Section 5, followed by a conclusion and directions for future work in Section 6.

2 Workflow modelling

Standard workflow modelling techniques distinguish between tasks, conditions and cases. A *task* refers to an indivisible piece of work that needs to be done. The order of these tasks is determined by *conditions*. The thing that is produced or modified as a result of the work carried out, is called the *case*.

In crisis management, the case is the incident or crisis that is being handled, e.g. a *fire* reported at the fire station. A condition represents the current state of the incident, for example whether the *fire* is being fought by firemen or not. The tasks in this example are the pieces of work carried out by the firemen, such as moving to the disaster area, or extinguishing the fire. In the CM-workflows discussed in this paper, all tasks are carried out by the same resource. We refer to this resource as the *actor*.

Workflows can be specified using Petri nets. Figure 1 specifies the workflow of the fire example.

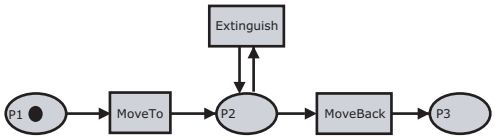

Fig. 1. The CM Workflow

A Petri net consist of places, transitions and tokens. Places are represented by ovals and correspond to the conditions of the workflow. Transitions are represented by rectangles and correspond to tasks. Tokens are represented by black dots and correspond to the cases being handled.

In Petri nets, transitions are the active components, i.e. they can move tokens from input places (places connected by incoming arrows) to output places (places connected by outgoing arrows). A transition can fire if a token resides at all of its input places. After a transition has fired, it consumes a token at each of its input places and produces a token at each of its output places.

In Figure 1, the transition *MoveTo* can fire, because place *p1* contains a token. After the transition has fired, place *p1* is empty, and place *p2* contains one token. In this configuration, the transitions *Extinguish* and *MoveBack* can fire (it is undefined which one of the two will actually fire). The process ends when the token arrives at place *p3*.

To model the characteristics of a case, known as case attributes, we extend the classical Petri net with *color*. In colored Petri nets, tokens have a value which can be used in conditional statements at transitions and which can be altered by the firing of transitions. For example, suppose that the token in Figure 1 contains the value ⟨*fire:on , location:townhall*⟩[1]. The transition *MoveTo* contains a conditional statement that only tokens with value *fire:on* can be consumed. This establishes that the firemen only move

[1] For readability, we have not included color information in the Petri net diagrams.

to locations where a fire is burning. The transition *MoveBack* states that the token must have the value *fire:out*. This prevents that the firemen leave the disaster area too quickly.

Contrary to most WM techniques, token values do not represent what the characteristics of the case actually *are*, but what the actor *knows* about the case. For example, the transition *Extinguish* states that after the transition has fired the token has the value *fire:unknown*. This represents that the firemen do not know whether their extinguishing efforts have resulted in the fire going out. It may be that there are still flames inside the building which are not visible from outside. For the fireman to continue with the task *Extinguish* or *MoveBack*, he must know whether the fire is still burning or not. In a ubiquitous computing environment, the fireman obtains this information via his PDA which establishes a wireless connection with the fire sensors in the building. For a technical analysis on the interaction between the PDA and the sensors in the environment, the reader is referred to [19].

In this paper, we are mainly concerned with the interaction between the PDA and the crisis worker. In the next section, we discuss different ways in which the PDA can present information to its user and how this can be modeled in the workflow.

3 Information Supply Mechanisms

To model an information supply mechanism, two additional Petri nets must be introduced. For modularity, we have separated these Petri nets from the main workflow. Additionally, the relations between these Petri nets and the main workflow are specified.

One Petri net model concerns the world, which is the ultimate source from which information is obtained. Figure 2 shows a model of a dynamically changing world. This Petri net simply replaces the value of the token in place *p15* with a random value, following either transition *Update1* or *Update2*. This simple model is sufficient for our purposes, but can be easily replaced by a more sophisticated world model, if required.

Fig. 2. World model

Another Petri net is used to model how information is obtained from the world and how the interaction protocol provides access to this information. Throughout the rest of this section, we will describe several simple models of well-known interaction mechanisms, such as query and subscribe.

Most insight into the information supply mechanisms is provided by the way in which the three individual Petri nets are combined into a whole. This aspect forms the most important part of the remainder of this section. For the query mechanism, this is described in Section 3.1. Three different kinds of subscribe mechanisms are described in Section 3.2, and a conditional subscribe mechanism is presented in Section 3.3.

3.1 Query

The first interaction mechanism we will discuss is Query. The left-hand side of Figure 3 shows the workflow plus one additional transition to establish the coupling with the query protocol. The places and transitions belonging to the workflow are colored grey. The right-hand side of the figure shows the query protocol. The two Petri nets are coupled by a so-called hierarchical transition (indicated by a double-lined box). The hierarchical query transition in the WF-net achieves that place p2 (i.e. the place with which the hierarchical transition is connected), becomes identical with the input/output place of the query Petri net (place *p4* which is labelled with "I/O").

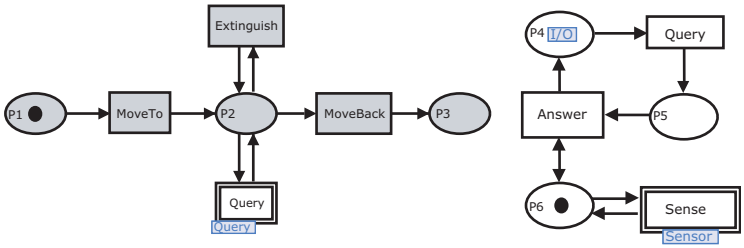

Fig. 3. Query

The mechanism works as follows. If the token arrives at place *p2*, both the transition *Query* and the transition *Extinguish* are enabled. If the transition *Query* fires, the token leaves the workflow net and arrives at place *p5*. The token stays there until the transition *Answer* fires. This transition produces a token at place *p4* (or equally well *p2*) with some of the token attributes replaced by the attributes of the token in *p6*. The token in *p6* represents the current sensor reading, which is occasionally updated by the hierarchical transition *Sense*. The Petri net behind this transition is the world model depicted in Figure 2. In this way, the token in *p6* is regularly updated with up-to-date world information. Note that the token in *p6* is not removed when the *Answer* transition fires, as it is connected with a double-headed arrow (it is both an input and an output place).

We will characterize this information supply mechanism by discussing interruption and optionality.

As appears from the Petri net specification, the mechanism causes a major interruption of the main workflow process. Firstly, the task *Query* must be performed, which is not a part of the workflow process. After that, the token arrives at place *p5*, which is also not part of the workflow process. It must wait until the transition *Answer* fires to return to the main workflow.

The other issue is that the obtaining of information is optional. This is because in place *p2*, two transitions may be enabled at the same time, i.e. *Extinguish* and *Query*. Hence, the fireman may choose not to ask for an information update and carry out

the *Extinguish* immediately. The benefit may be time savings. The drawback may be ignorance.

3.2 Subscribe

Besides querying, a common interaction mechanism in ubiquitous computing and peer-to-peer systems is the Publish-Subscribe protocol [15]. By subscribing to a piece of information, a continuous flow of information is initiated. There are three ways in which this information may actually be absorbed by the actor: non-interruptive,non-optional; interruptive,optional or interruptive,non-optional.

Figure 4 specifies the non-interruptive, non-optional subscribe mechanism. One can think of this subscribe mechanism as the low fuel light on the dashboard of a car. It guarantees that information is delivered to the driver (it is non-optional) and does not require any effort (its is non-interruptive).

In the Petri net specification, the workflow model (on the left) is coupled with the subscribe model (on the right) using a fusion set. [2] (called *Fusion 1*). Multiple places that occur in the same fusion set become identical. The mechanism works as follows. The subscribe protocol contains a token in place *p6* which represents the content of the subscription, for example *fire:on*. If this content matches the current sensor reading (represented by the token in *p7*), the transition *Notify* fires. Otherwise, the transition *Remain Silent* fires. The token in place *p5* (and likewise *p3*) is provided with up-to-date token attributes as a result of this firing. The information arrives at the actor during the execution of his tasks, i.e. the information in the token of *p3* is blended with the information in the token of the workflow.

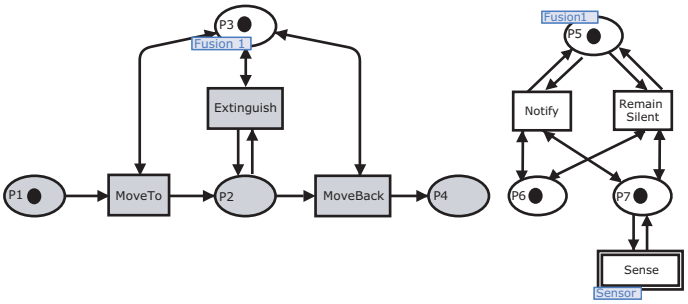

Fig. 4. Subscribe 1 (non-interruptive, non-optional)

Contrary to the Query mechanism, this communication mechanism does not cause any interruption. This is expressed in the Petri net specification by the fact that the case

[2] The difference of assembling Petri nets with a fusion set instead of with a hierarchical transition, is that when the hierarchical transition occurs multiple times in the main Petri net, multiple instances of the sub-Petri net are created. This is not the case when places from the same fusion set occur multiple times in the main Petri net.

token can never leave the workflow model. Furthermore, the obtaining of up-to-date information is not optional but is enforced by the model.

Some information is too complex to be conveyed without interruption as it requires some mental processing by the receiver. For these cases, an interruptive subscribe mechanism can be used. Figure 5 shows an interruptive, optional subscription mechanism. One can think of this subscribe mechanism as the clock on a mobile phone. The phone maintains up-to-date information which the owner can choose to consult by investing a little effort, namely getting it out of his pocket.

In the specification *p4*,*p5*,*p6* and *p7* are identical, and contain a token which represents up-to-date information about the world. When the transition *Consult* fires, the token attributes of this token are passed to the token in the workflow model.

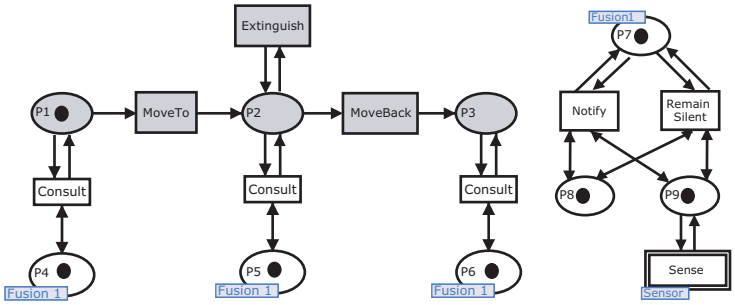

Fig. 5. Subscribe 2 (interruptive, optional)

This interaction mechanism causes a little interruption. This appears from the fact that the *Consult* task (which is not part of the workflow model) must be carried out. As with the Query mechanism, the obtaining of information is optional.

Yet another type of subscribe mechanism is the interruptive, non-optional subscription, as depicted in Figure 6. This mechanism can be thought of as a mobile phone, i.e. when it rings, the owner is forced to consult it before he can continue with the task at hand.

This specification uses two fusion sets. *Fusion 1* consists of *p4*,*p6*,*p8* and represents that there is currently a notification to be processed. *Fusion 2* consists of *p5*,*p7* and *p9* and represents that there is currently no notification to be processed. The notification protocol (shown on the right of the figure) ensures that a token cannot be in the groups *Fusion 1* and *Fusion 2* at the same time. This is because, when *Notify* fires, the token is removed from *Fusion 2* and added to *Fusion 1*. The only way that the token can return to *Fusion 2* is via *Consult*, which removes a token from *Fusion 1* and adds a token to *Fusion 2*. Therefore, when a token resides in *Fusion 1* the tasks in the workflow are blocked (because all tasks require a token to be present in *Fusion 2*). This means that the actor has no choice but to consult the notification. After the transition *Consult* has fired the token is moved from *Fusion 1* to *Fusion 2*, and the workflow tasks are enabled again.

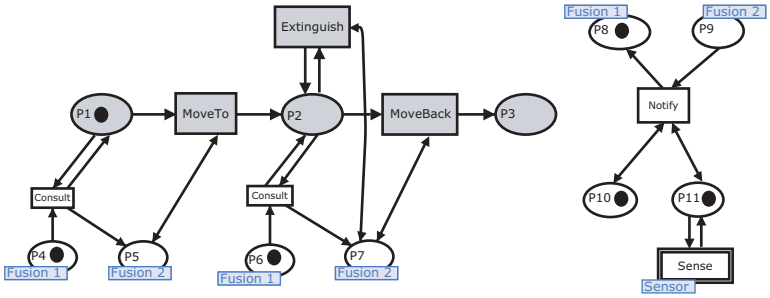

Fig. 6. Subscribe 3 (interruptive, non-optional)

Like the previous subscribe protocol we discussed, this mechanism causes a little interruption due to the *Consult* task. However, this subscribe mechanism is not optional, i.e. when a notification is sent, the actor has no choice but to turn his attention to it.

3.3 Conditional Subscribe

By using a conditional subscription, a user requests to receive notifications of something only if some condition holds. For example, the fireman may request to receive notifications about *fire:out* only when he is at the disaster area, and not when he is at the office. In order to realize such a notification system, the system must be *context-aware* [1], i.e. it must know the user's location and adapt its behavior to it. Although for every communication mechanism discussed so far, a context-aware variant can be specified, we focus on a variant of the interruptive, non-optional subscribe mechanism (Subscribe 3 in Figure 6). As an example of this conditional subscribe mechanism, one can think of a context-aware mobile phone, which automatically shuts off when the user enters a lecture hall.

The specification of the conditional subscribe is depicted in Figure 7. This specification uses a third fusion set *Fusion 3*, consisting of *p13*, *p14* and *p15* (see the world model in Figure 2). The purpose of this fusion set is to model the effect of task execution on the world (place *p15*). For example, the transition *MoveTo* updates the token in place *p13* (and *p15*) to represent the information that the fireman is no longer at the office but at the disaster area. The *Sense* transition is specified such that it also obtains information about the location of the fireman (for example, using GPS). An extra place is added (*p12*), which represents the condition when the subscription applies (in our example, that the fireman is not at the office). Like the protocols discussed before, place *p10* represents the information to which the actor is subscribed (in our example *fire:out*).

With respect to interruption and choice, this protocol has the same properties as the third subscribe protocol (Figure 6). It causes a little interruption when a notification is received and the owner of the device has no choice but to process the incoming notifications.

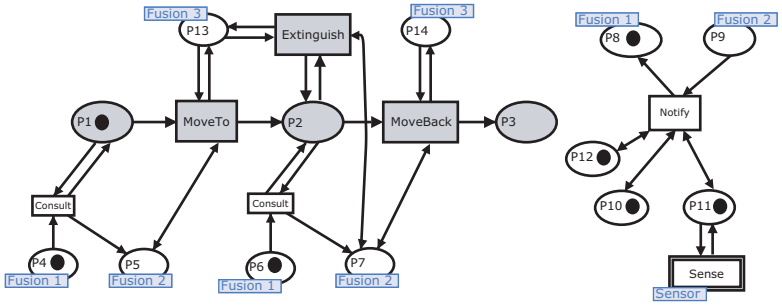

Fig. 7. Conditional Subscribe (interruptive, non-optional)

3.4 Comparison

The differences between the five information supply mechanisms discussed in this section are summarized below. We have used three degrees to indicate interruption. High interruption (+) means that a transition and a place are visited which are not part of the workflow model. Medium interruption (+/-) means that only a transition is executed which is not part of the workflow model. No interruption (-) means that no states and transitions are executed which are not part of the workflow. Optionality can be positive (+), when the actor has a choice to obtain information, or negative (-) when there is no such choice. Context aware (+) means that the information supply mechanism behaves differently depending on at which place the actor resides at that moment. Otherwise, the system is not context aware (-).

	Query	**Subscr1**	**Subscr2**	**Subscr3**	**CondSubscr3**
Interruption	+	−	+/−	+/−	+/−
Optionality	+	−	+	−	−
Context aware	−	−	−	−	+

As we have argued before, a context aware variant can be made also for the other information supply mechanisms. This would yield three more protocols. With respect to the properties of Interruption and Optionality, there are six possible combinations. We believe that the four combinations we have covered are the only sensible ones. A protocol with optionality and no interruption is impossible because a choice can only be modeled by introducing a transition which is not part of the workflow, which would cause interruption again. A protocol with high interruption but no optionality would be possible, but not very useful, because the second subscribe mechanism can be used for the same purpose, causing only medium interruption.

4 Analysis Techniques

We have implemented the Petri nets described in this paper in CPN-tools [5], a tool for modelling and validating colored Petri nets. Using this tool, several kinds of errors in the model can be detected.

Firstly, there are the trivial structural errors in the design model. The transitions and tokens may be wrongly connected, the transitions may be configured in a way which conflicts with the token values, or the data structures may contain syntactic errors. In these cases, the CPN-tool will simply generate a syntax error stating that the Petri net is incorrect.

More interesting are the errors in the crisis management model which the Petri net represents. For the workflow part of the model, all analysis techniques can be used which have been developed for workflow analysis [17]. For example, it can be checked whether the workflow terminates, i.e. whether the token eventually arrives at a place which is the final destination (in our example, this is place *p3*). Another important property for workflow Petri nets is the boundedness property. This property states that every place will never contain more than a certain number of tokens. This is important because otherwise the relation between the token and the case that it represents becomes unclear.

In the remainder of this section, we describe some validation criteria that are specific to the combination of workflow models and information supply mechanisms.

The first issue we discuss is information underload. In our workflow model, we have represented the information needs of the actor [18] as the *preconditions* of a task. For example, the transition *MoveBack* in Figure 1 can only fire if *fire* has the value *out*. Because the precondition of the task *Extinguish* is *fire:on*, a token with value *fire:unknown* in *p2* can go nowhere. In Petri net theory, such a situation is called *deadlock*. Many algorithms exist to compute deadlock situations efficiently. In our crisis management models, deadlock indicates information underload of the actor, which should be resolved by adding information supply mechanisms. For example, the workflow model in Figure 1 suffers from information underload. This is because after the transition *Extinguish* has fired the value of *fire* is *unknown* and the net is in a deadlock situation. The other models (in Figure 3, 4, 5, 6, 7), do not lead to a deadlock situation and thus do not suffer from information underload.

As mentioned before, another important problem in crisis management is cognitive overload, i.e. when the actor receives too much information to be able to stay focussed on his main tasks. In our framework, we can measure the expected cognitive load of the information supply mechanisms by performing a simulation analysis. Using CPN-tools, the execution of a Petri net can be simulated to analyze how the token moves through the transitions and places. By applying a counter to some of the transitions and places that do not belong to the main workflow (such as *consult* in our example), an indicator is obtained on the expected cognitive load.

We have applied this simulation analysis to the Petri-nets described in this paper. Because the workflow and world model are very simple, this analysis merely confirmed the properties which we had already theoretically determined in Section 3.4. In a more realistic scenario, however, the workflow is much larger and the actor might obtain his information from multiple sources which makes it impossible to theoretically foresee all possible behaviors of the model. In these cases, a simulation analysis provides a valuable contribution to what can be theoretically assessed.

5 Related Work

We will divide our discussion on related work into three categories, corresponding to the different purposes for which our approach can be used.

Firstly, our approach can be viewed as a formalization of an interaction protocol. Petri-net formalizations of computer-computer interaction have a long tradition in computer science (e.g. [9], [2]). However, theories on human-computer interaction are usually not mathematically formalized. For example, [8] identifies an *attention-utility trade-off* in notification systems. The costs of notification are defined as the amount of attention removed from the user's primary task. The benefits of a notification system are characterized along three dimensions, i.e. comprehension, reaction and interruption. Another approach addressing this trade-off focusses on peripheral information displays [7]. This work presents experimental results on what we would call a non-interruptive, non-optional subscribe mechanism (Subscribe 1). Whereas these approaches nicely identify the different aspects that are important in human-computer communication, they do not provide a formal underpinning. We believe that our Petri-net formalization of human-computer communication is useful to make the attention-utility trade-off more precise, resulting in a better understanding of the different aspects involved.

A second way to view our approach is as a method for analysing and evaluating interactive systems. For this purpose, task models are frequently applied, e.g. ConcurTaskTrees [14]. Whereas task models and workflows are tightly related [6], there are also some differences. We will discuss some relevant correspondences and differences between ConcurTaskTree models and Petri-net workflow models below. Both models allow the representation of concurrency, have an intuitive graphical syntax, and can be automatically verified. A difference is that task models are usually hierarchically structured whereas workflows are not. Because for our purposes, it suffices to view the work process at one level of abstraction, this restriction of workflow models is not problematic. Another difference is that Petri-nets are suitable for modelling information flows at a high level of detail (as proven by the popularity of Petri-net based communication protocols), whereas it is not clear how this can be done in a task model. Because modelling information flows is crucial to our approach, we have chosen for Petri-net workflows instead of ConcurTaskTree models.

A third way to apply the results described in this paper is as a model to be used by the computer at runtime (as is proposed in [13]). For example, it can be used by a PDA to adjust its notification style by estimating the cognitive load it imposes on its user. Because our model is computational, it can be applied to this purpose.

6 Conclusion and Future Research

In this paper, we have characterized different information supply mechanisms by specifying their relation with a workflow. This allows us to precisely capture those aspects of query and subscribe mechanisms that are important for human crisis workers. In general, we believe the techniques proposed in this paper provide valuable insights for modelling the interaction between humans and multi-agent systems.

We have identified two promising directions for future research. Firstly, we plan to apply more advanced workflow modelling techniques to enable a more thorough analysis of crisis management. One option would be to extend the Petri nets with time. This would allow us to estimate the average completion time of the process, and to study the influence of different information supply mechanisms on this. Another option would be to use *adaptive workflows*, which is the area of workflow management concerned with modelling exceptions on the normal course of action. Particularly for crisis management, this aspect is highly relevant.

Another direction for future research is concerned with agent-organizational aspects of workflow modelling [12]. It is typically unknown at design time which sensors will be available at the time and place a crisis takes place. Therefore, it should be possible to discover and invoke sensors that provide valuable information at runtime. In workflow modelling, this is called resource allocation, i.e. assigning a resource to a task in an efficient way. In our case, the resource is a sensor and the task is making a measurement. We plan to investigate how results from resource allocation can be used to enhance efficiency in ubiquitous computing, for example to decide which sensor can best be queried for which type of information.

References

1. G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles. Towards a better understanding of context and context-awareness. pages 304–307, 1999.
2. R. S. Cost, Y. Chen, T. Finin, Y. Labrou, and Y. Peng. Using colored petri nets for conversation modeling. In F. Dignum and M. Greaves, editors, *Issues in Agent Communication*, pages 178–192. Springer-Verlag, 2000.
3. T. Finin, R. Fritzson, D. McKay, and R. McEntire. KQML as an agent communication language. In *Proceedings of CIKM*, pages 456–463. ACM Press, 1994.
4. G. Fischer. User modeling in humancomputer interaction. In *User Modeling and User-Adapted Interaction*, volume 11. Springer, 2001.
5. K. Jensen, L. Kristensen, and L. Wells. Coloured petri nets and CPN tools for modelling and validation of concurrent systems. *International Journal on Software Tools for Technology Transfer*, 9(3):213–254, 2007.
6. R. Kristiansen and H. Traetteberg. Model-based user interface design in the context of workflow models. In *Task Models and Diagrams for User Interface Design*, volume 4849 of *LNCS*, pages 227–239. Springer, 2007.
7. P. P. Maglio and C. S. Campbell. Tradeoffs in displaying peripheral information. In *CHI '00: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 241–248, New York, NY, USA, 2000. ACM.
8. D. S. McCrickard and C. M. Chewar. Attuning notification design to user goals and attention costs. *Commununications of the ACM*, 46(3):67–72, 2003.
9. K. Mikkilineni, Y.-C. Chow, and S. Su. Petri-net-based modeling and evaluation of pipelined processing of concurrent database queries. *IEEE Transactions on Software Engineering*, 14(11):1656–1667, 1988.
10. M. Neerincx and E. Griffioen. Cognitive task analysis: harmonizing tasks to human capacities. *Ergonomics*, 39(4):543 – 561, 1996.
11. N. Netten, G. Bruinsma, M. van Someren, and R. de Hoog. Task-adaptive information distribution for dynamic collaborative response. *Special Issue on Emergency Management Systems of the International Journal of Intelligent Control and Systems (IJICS)*, 11(4):237–246, 2006.

12. M. D. Oliveira, S. Cranefield, and M. Purvis. Normative spaces in institutional environments by the means of commitments, reputation and colored petri nets. In *Proceedings of the 8th International Workshop on Agent Oriented Software Engineering (AOSE)*, 2007.
13. S. Pangoli and F. Paternó. Automatic generation of task-oriented help. In *UIST '95: Proceedings of the 8th annual ACM symposium on User interface and software technology*, pages 181–187, New York, NY, USA, 1995. ACM.
14. F. Paternò, C. Mancini, and S. Meniconi. Concurtasktrees: A diagrammatic notation for specifying task models. In *INTERACT '97: Proceedings of the IFIP TC13 Interantional Conference on Human-Computer Interaction*, pages 362–369, London, UK, UK, 1997. Chapman & Hall, Ltd.
15. A. Ranganathan and R. H. Campbell. An infrastructure for context-awareness based on first order logic. *Personal Ubiquitous Computing*, 7(6):353–364, 2003.
16. W. van der Aalst and K. van Hee. *Workflow management: models, methods, and systems*. MIT Press, Cambridge, MA, USA, 2002.
17. W. M. P. van der Aalst. Verification of workflow nets. In *ICATPN '97: Proceedings of the 18th International Conference on Application and Theory of Petri Nets*, pages 407–426, London, UK, 1997. Springer-Verlag.
18. J. van Diggelen, R. J. Beun, R. M. van Eijk, and P. J. Werkhoven. Modeling decentralized information flow in ambient environments. In *Proceedings of ambient intelligence developments (AmI.D '07)*, 2007.
19. J. van Diggelen, R. J. Beun, R. M. van Eijk, and P. J. Werkhoven. Agent communication in ubiquitous computing: the ubismart approach. In *Proceedings of the Seventh International Conference on Autonomous Agents and Multi-agent Systems (AAMAS'08)*, pages 813–820. ACM, 2008.
20. M. Weiser. The computer for the 21st century. *Scientific American*, 265(3):66–75, 1991.