A Survey of Task Computing

Chuong C. Vo, Torab Torabi, Seng W. Loke

Abstract

1 Introduction

2 What is Task-Driven Computing?

According to Masuoka, Labrou, Parsia and Sirin (2003); Wang and Garlan (2000), *task-driven computing* or *task computing* allows users to focus on the tasks they want to accomplish rather than how to accomplish them.

To transfer users' focus from the computer to the task at hand. In other words, to help a person "forget he/she is using a computer while interacting with one" Coen (1998).

2.1 Challenges

1. Task computing should support for task migration. That is a capability to suspend task execution from one environment and resume it later in a different environment.

3 What is a Task?

3.1 Definitions of Tasks

There are many definition of a task in the literature (e.g., Paternò (2001); Ranganathan and Campbell (2005); Rich (2009); Wang and Garlan (2000); **?**) but there is no a common definition of a task. We adopt a definition that "*a task is a goal or objective which is presented by a set of actions performed collaboratively by humans and machines to achieve this goal*".

3.2 Tasks vs. Services

A task is different from a service. Services are means to accomplish tasks. A task is a goal or objective while a service is the performance of tasks. Tasks are associated with what the user wants to accomplish, and services with environmental capabilities to complete the tasks **?**.

"Task" is a high-level and user-oriented term, which is associated with user's requirements, while "service" is a low-level and system-oriented term, which is associated with system functionalities.

3.3 Tasks vs. Problems

A task is not the same concept as a problem.Mizoguchi et al. (1995) This statement is justified by the fact that one can say "perform a task" but would not say "perform a problem", which shows their inherent difference. A task is as a sequence of problem solving steps. Therefore, the name of a task necessarily includes verbs representing problem solving activities.

3.4 Characteristics of Tasks

Tasks vary widely in their time extent. They can last minutes, hours, or days. Some are effectively instantaneous, for example, 'turn off the lights'; some never finish; and some have an indeterminable time extent. They typically involve both human participants-as requesters, beneficiaries, or performers of the tasks-and machines. Some tasks can be performed only by a human being; others can be performed only by a machine; and yet others can be performed by either. Tasks can span multiple devices, places, and involve many people, services. They can range from a very high abstraction level (such as 'contact someone') to a concrete, action-oriented level (such as 'enter phone number'. Very low-level tasks are closer to the primitive controls of a particular device, for example, 'tap the Call button on the phone'. Finally, tasks are often personalised for each user.

4 What is a Task Model?

A model of a task (called task model, task description, task specification, task formalism, or task expression) is a formal description of the activities/actions/sub-tasks involved in completing the tasks Paternò (2001); Rich (2009). A description of a task represents the decomposition of the task into its subtasks/actions. When we reach (sub)tasks which cannot be further decomposed we have basic (atomic or primitive) tasks. In some cases, basic tasks require one single physical action to be performed.

In the past, a task model was a means to help designers, developers, managers, customers, and experts understand how the task should be performed. Specifically, it helps designers identify requirements which should be satisfied to perform the task effectively. It also helps a machine understand how the task are performed Mori et al. (2002).

Smart spaces are highly dynamic environments in terms of the availability and heterogeneity of users, devices, and services. Therefore, the key requirements for a language used to describing task models are to support for the abstraction of task models, i.e., platform-independence. Task models would be abstract enough to avoid being bound to specific platforms. The abstraction of task models allows tasks to be flexibly executed in different environments. So, this would increase the reuse and scalability of task models.

5 Task-Driven Computing Framework

A task-driven computing framework or task computing framework is to deal with mapping between tasks and services. For example, a lecture hall fitted with a task computing framework can process typical tasks, such as showing presentations and controlling lighting, by completing a set of underlying services **?**.

6 Taskable Spaces

A spaces fitted with a task computing framework are taskable space **?**.

7 Task Ontologies

"Roughly speaking, an ontology is composed of two parts, that is, taxonomy and axioms. Taxonomy is a hierarchical system of concepts while axioms are established rules, principles, or laws among the concepts. Axioms specify the competence of an ontology. In other words, a class of the questions to which the answers can be derived from the axiom specifies the competence of the ontology." Ikeda et al. (1998)

Tasks, which are commonly performed at a place or with particular appliances, yields a vocabulary of typical task sets, called a *task ontology* **?**. A task ontology provides a common sense of the tasks which places or artifacts should support. For examples, in a home theater room, a common task would likely be watching a movie; in a lecture hall, a typical task is such as continue yesterday's lecture; consider windows and drapes–typical tasks are open and close; typical tasks on lights are turn on and turn off; and if a dimmer is present, darker or brighter. **?**

A task ontology could be built on a situation ontology. A situation ontology is built on a time ontology, a place ontology, an artefact ontology, a user ontology, an environment condition ontology...

Mizoguchi Mizoguchi et al. (1995) defines a task ontology as a system of vocabulary for describing problem solving structure of all the existing tasks domain-independently. It is obtained by analysing task structures of real world problems. The ultimate goal of task ontology research includes to provide vocabulary necessary and sufficient for building a model of human problem solving processes. When we view a problem solving process based on such as a sentence of natural language, a task ontology is a system of semantic vocabulary for representing meaning of the sentence. A task ontology consists of the following four kinds of concepts:

1. Generic nouns representing objects reflecting their roles appearing in the problem solving process,

2. Generic verbs representing unit activities appearing in the problem solving process,
3. Generic adjectives modifying the objects, and
4. Other concepts-specific to the task.

8 Task-Oriented User Interfaces

A task-oriented user interface (task-based user interface) of a smart space are a user interface which are oriented (navigated) around the tasks that a user have for the space. "For example, a reasonable goal for the user after opening the freezer is to want to defrost something. Thus if a system can sense the door opening, the microwave should suggest the defrosting function. If the devices cannot perform a desired action, they should give the user an explanation of why the goal failed and how to fix it. If the state of the device interferes with another action, it should inform the user of the conflict and the actions necessary to resolve it (Self-describing). In the normal process of manipulating the devices, the user may be required to make choices in situations where he or she might not have a good understanding of the consequences. In this case, the system should inform the user about the trade-off of each choice. Informing the user of the consequences of trade-offs has the following advantages: (a) it prevents the user from experiencing an undesirable and potentially irreversible consequence of a system action; and (b) it helps the user back trace to the right point if he or she wants to change the behaviour of the system (Self-revealing). Fixing devices when things go wrong is one of the most frustrating things about dealing with consumer electronics. A common reason for problems is when the users use of the device is outside the designers anticipated scenario of use, or there is a piece that it is not working as expected. Fixing this problem forces the user to introspect about the systems internal state which might be hidden by the device designer and to figure out what is wrong and devise a strategy to fix it (Self-debugging)."

9 Task-Focused Interfaces

9.1 Introduction

According Wikipedia[1], a *task-focused interface* is a type of user interface which extends the desktop metaphor of the graphical user interface to make tasks, not files and folders, the primary unit of interaction. Instead of showing entire hierarchies or lists of information, such as the contents document hierarchy, a task-focused interface shows the subset of the content that is relevant to the task at hand. This addresses the problem of information overload when dealing with large hierarchies, such as those in software systems or large sets of documents. The task-focused interface is composed of a mechanism which allows the user to specify the task being worked on, a model of the task context such as a degree-of-interest (DOI) ranking Leung and Apperley (1994), and a mechanism to filter or highlight the relevant documents.

[1]http://en.wikipedia.org/wiki/Task-focused_interface

9.2 Methods

Based on the user's interaction with information, each uniquely identifiable element of information available to the user is assigned a degree-of-interest (DOI) ranking. The more frequently and recently a user has interacted with an element of information, the higher the DOI for that element for that task.

The DOI rankings for the information elements can be used within a task-focused interface in four ways. Elements below a certain DOI threshold can be filtered to reduce the number of elements presented. Elements can be ranked according to their DOI; for instance, the elements of highest interest can be shown at the top of a list. The elements can be decorated with colours to indicate ranges of DOI. Finally, the display of structured information elements can be automatically managed based on DOI; for instance, text corresponding to elements with low DOI can be automatically elided.

The DOI value for each information element interacted with as part of a task can be derived from a stored history of interaction events recorded as the user works with the application. This approach requires a user to indicate the start of a task. The collection of all interaction events that take place during a single task is called a "task context".

9.3 Prototypes

The Eclipse Mylyn project[2] is an implementation of the task-focused interface. Mylyn filters, sorts, highlights, folds, and manages tree expansion for numerous views within the Eclipse IDE based on the currently active task.

Tasktop Pro[3] is a full-featured supported product based on Mylyn with additional productivity features.

Task-based service navigationFukazawa et al. (2005, 2006); Luther et al. (2008); Naganuma and Kurakake (2005); Naganuma et al. (2006); Sasajima et al. (2007, 2006).

Task-oriented user interface to a digital libraryCousins (1996); Meyyappan et al. (2004).

Articulating the Task at Hand and Making Information Relevant to ItFischer (2001).

10 Task Computing

10.1 Introduction

Task computing: "shifting focus to what users want to do (i.e., on the tasks at hand) rather than on the specific means for doing those tasks".Masuoka, Parsia and Labrou (2003) Task Computing as computation to fill the gap between tasks (what user wants to be done), and services (functionalities that are available to the user).Masuoka, Parsia and Labrou (2003) "The fundamental premise of task computing is to present to the user the tasks that are possible in the user's current context, to assist the user in

[2]http://www.eclipse.org/mylyn/index.php
[3]http://tasktop.com

creating more complex tasks by using simpler tasks as building blocks and to guide the user through the execution of the complex tasks. Complex tasks may span multiple applications and multiple computing platforms (not a single device)."

"Task" is a high-level and user-oriented term, which is associated with user's requirements, while "service" is a low-level and system-oriented term, which is associated with system functionalities. Pan et al. (2011)

The concept of tasks, which might also be called activities, goals, jobs, or actions. Rich (2009) For examples,

A task model is a plan to achieve the goal of the task.

A task model defines task classes. A task instance corresponds to an actual or hypothetical occurrence of a task class. Pressing the power button on a DVD player is an example of a task class. But David Smith pressing the power button on the DVD player in his living room at 3:15pm on 1^st January 2006 is an instance of this class. For example taken from the overall example task model description.

Spontaneity: coordinating available services during runtime, without having a previous definition of applications.

"the task-based interface model, in which users can arrange their machine's entire interface architecture to present only those applications and artifacts pertaining to a particular task." Goth (2009)

"Transition from essentially unimodal, menu-based dialogue structures (with a fixed interaction vocabulary provided by the system) to polymodal, conversational dialogue structures (with an unrestricted interaction vocabulary provided by the user). Transition from a function-oriented interaction with devices to a goal-oriented interaction with systems." Heider and Kirste (2002)

A framework for resource discovery in pervasive computing for mobile aware task execution

A task migration framework based on web services in ubiquitous computing environment

A task-oriented approach to support spontaneous interactions among users in Urban Computing environments

A task-driven user interface architecture for ambient intelligent environments

A user-centric task computing architecture for pervasive computing

An activity-centered ubiquitous computing framework for supporting occasional human activities in public places

An agent based platform for task distribution in virtual environments

An Architectural View of the Entities Required for Execution of Task in Pervasive Space

Aura: an architectural framework for user mobility in ubiquitous computing environments From Computers Everywhere to Tasks Anywhere: The Aura Approach Beyond Desktop Management: Scaling Task Management in Space and Time

Automatic Execution of Tasks in MiPeG

Castaway: a context-aware task management system

Intelligent user task oriented systems

InterPlay: A middleware for integration of devices, services and contents in the home networking environment

Integrating Place and Time with Tasks: Adding Transportation to PIM

Mobile adaptive tasks guided by resource contracts

Middleware for management of end-user programming of clinical activities in a pervasive environment

Nimbus: A task aware/context aware mobile computing platform

Operating system support for task-aware applications

PerFlows for the computers of the 21st century

Plan-Driven Ubiquitous Computing

Self-optimization of task execution in pervasive computing environments

Sentient Processes - Process-based Applications in Pervasive Computing Experience using processes for pervasive applications

Software Architectures for Task-Oriented Computing

Task computing for ubiquitous multimedia services

Task computingthe semantic web meets pervasive computing

Task Follow-me: Towards Seamless Task Migration Across Smart Environments

Task-based design and runtime support for multimodal user interface distribution

The benefits of embedded intelligence - Tasks and applications for ubiquitous computing in logistics

Towards Ubiquitous Task Management

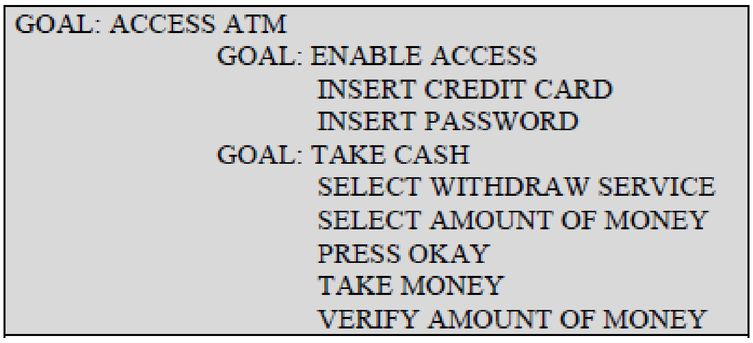
11 Task Model Description Languages

11.1 CEA-2018 Task Model Description–CE TASK

A task description language provides a formal syntax and semantics for creating task models. The constructed models can then be used to specify and communicate interface designs, generate interfaces, predict the usability of interfaces, or enable systems to monitor user activities.

There are many approaches on task modelling. Rich (2009) points out the main disadvantage of the traditional approaches that task models are used only for user interface design at design time, if at all, then discarded. The next disadvantage is the device-dependance of task models. This limits the scalability of task models.

Most existing approaches focused on tasks which are either a desired modification of the state of an application or an attempt to retrieve some information from an application (ex., Accessing a flight's database to know what flights are available is a task which does not require the modification of the state of the application, whereas Accessing a flight's database to add a new reservation requires a modification of the state of the



```
GOAL: ACCESS ATM
        GOAL: ENABLE ACCESS
            INSERT CREDIT CARD
            INSERT PASSWORD
        GOAL: TAKE CASH
            SELECT WITHDRAW SERVICE
            SELECT AMOUNT OF MONEY
            PRESS OKAY
            TAKE MONEY
            VERIFY AMOUNT OF MONEY
```

Figure 1: Example of a GOMS specification.

application). Now, we have a task which aims to change the state of the environment.

11.2 Hierarchical Task Analysis

Hierarchical Task AnalysisAnnett and Duncan (1967): logical hierarchical structure

11.3 GOMS

"GOMS (Goals, Operators, Methods, Selection rules)Card et al. (1983) provides a hierarchical description to reach goals in terms of operators. Operators are elementary perceptual, motor and cognitive acts. Methods are sequences of subgoals and operators used to structure the description of how to reach a given goal. The selection rules indicate when to use a method instead of another one.

In Figure 1, there is an example of a GOMS specification. It describes how a user segments the larger goal of Access ATM into a sequence of small, discrete operators, such as Select withdraw service or Select amount of money. In this example no selection rule is used."

as means for formally describing human problem solving behaviour.

11.4 Grammar-Based Approaches

Task Action GrammarPayne and Green (1986). "In these grammars the logical structure is defined in the production rules where high level tasks can be associated with non terminal symbols whereas basic tasks are associated with the terminal symbols of the grammar."

11.5 UAN

UAN (User Action Notation)Hartson and Gray (1992)'s main purpose is to communicate design. It allows designers to describe the dynamic behaviour of graphical user interface. It is a textual notation where asynchronous tasks are represented in a quasi-hierarchical structure. A UAN specification is usually structured into two parts:

- One part describes task decomposition and the temporal relationships among asynchronous tasks;
- The other part associates each basic task with one table. The table have three columns indicating the user actions, the system feedback and the state modifications requested to perform it.

11.6 LOTOSBolognesi and Brinksma (1987); ?

11.7 ConcurTaskTrees

ConcurTaskTreesPaternò et al. (1997) is a graphical language for modelling tasks. It provides a rich set of operators to describe the temporal relationships among tasks (enabling, concurrency, disabling, interruption, optionality). For each task further information can be given such as its type, the category (indicating how the performance is allocated), the objects that it requires to be manipulated and attributes, such as frequency of performance.This model has the following features:

- It is a hierarchical structure and graphical syntax.
- It allows to define temporal relationships between the tasks such as enabling, concurrent performance, choice, parallel composition, disabling, suspend-resume, order independence, independence concurrency, concurrency with information exchange, order independence, and enabling with information passing.
- It describes task allocation by indicating the category of the task including user task, application tasks, interaction task, and abstract task (tasks that have subtasks belonging to different categories). It also allows the specification of who or what is performing the task, whether it be the user, the system, or an interaction between the two.
- It also describes the objects including user interface objects and application domain objects that have to be manipulated to support their performance.
- It allows to specify the properties of a task such as iteration, finite iteration, optional task, and recurrence, preconditions, identifier, name, frequency of use, access rights, and estimated time of performance.
- Moreover, it allows to indicate cooperative tasks where two or more users are involved to perform the tasks. Using this model, a cooperative task is decomposed until the sub-tasks are performed by single users.

A graphical tool called CTTE Mori et al. (2002) was developed to support editing and analysis of task models. The noticeable features of the tool are the follows:

- Editing task models;
- Checking completeness of task models;
- Multiple interactive views of task models;

- Comparing task models;
- Task model simulator which can be used for interactive simulation of a developed task model.

The limitations of this approach:

- Task models using this approach only support for designing of applications which are used in statically target environments with static goals and specific devices. It is impossible to make changes to the task model at runtime to adapt the task performance to specific circumstances.
- Objects associated with tasks are insignificantly abstract so that they are not best suited in dynamic environments.
- This model does not allow the user to interrupt a task and then resume it later in a different environment (task migration). In highly dynamic environments, users would like the task follow them between different environments.
- does not have an operator defining the premature termination of a scenario (whether it is due to human or system error).

Sinnig et al. (n.d.) propose a set of extensions for these task models. extensions to the operator set (namely stop, instance iteration, non-deterministic choice, and deterministic choice.); structural extensions (A task model is no longer defined as a monolithic task tree but in a modular fashion where a task tree may include references to other subordinate task trees. define a specialisation relation between task models and propose a high-level notation called "Task Model Diagram".), and (3) extensions in support of cooperative task models (addresses the creation of task models for cooperative applications (e.g. multi-user smart environments). define a concept of a cooperative task model. Within such a cooperative task model the execution of a task of one model may enable or disable the execution of a task in a different task model, extend CCTT by taken into account that a role is typically fulfilled by several users. For each user we create a copy (instance) of the corresponding role task model. At runtime the various instances of the task model are executed concurrently. Synchronisation points between instances are specified in TCL (task constraint language). A coordinator task model, as specified in CCTT, is not needed.).

In order to overcome CTT's inability to specify task failures and error cases Bastide and BasnyatBastide and Basnyat (n.d.) introduce the concept of error patterns.

11.8 UIDL

The PUC's Specification LanguageNichols and Myers (2009) (UIDL) is designed for modelling functional models of individual appliances. UIDL does not include a task model.

11.9 UDIT

UDIT Meixner et al. (2009) is a graphical editor for useML.

11.10 The Room-Based Use Model

The Room-Based Use Model Breiner et al. (2009) is the extended Use Model.

11.11 Tasks as Virtual Services

Garlan et al. (2002); Masuoka, Parsia and Labrou (2003); Wang and Garlan (2000) simply treated a task as merely binding together a set of relevant computing applications (called services). The fundamental problem with this approach is that it is too application-centric. Since these applications are only a means to carry out a task, they are not suitable to represent the task itself, or to capture user-related context information.

In Project Aura Garlan et al. (2002); Wang and Garlan (2000), user tasks are represented as coalitions of abstract services that are at runtime bound to actual services in the environment. By providing an abstract characterisation of the services in a task, it is possible for a system to search heterogeneous environments for appropriate actual services to supply those virtual services. A task is simply a top-level virtual service that can be invoked directly by a user. A unit of abstract functionality is represented by a virtual service. A task is regarded as a top-level virtual service which may be decomposed into a set of virtual services. Thus a task can be achieved by executing its virtual services which are bound to physical services. The Aura project has no semantic modelling of tasks which makes it difficult in context-adaptively matching services in an open manner to continue a user task.

The interfaces of a virtual service include four elements: a functional interface, a state interface, a configuration interface, and a dependency specification. The functional interface defines how a client can access the service. It can be described using an ordinary interface definition language (such as CORBA IDL) in the forms of communicating messages, method invocations, and data pipes.

The state interface describes the state constituents of a virtual service. State is defined using a structure schema (such as an XML schema) that specifies the state of a service in terms of attributes and values. For example, the state schema of an email service might include attributes such as current mailbox, current message_ID, email address shortcuts, etc. The state schema of a document display service will include the name of the document, the current position in the document, and possibly some display options.

The configuration interface describes the configuration parameters of a virtual service. It is also represented by a structure schema. For example, an email service configuration interface might specify options, such as the mail server name; a document editing service may include options such as the auto-save interval and tab size.

The dependency specification can be used to specify other services required by a virtual service in order to operate. For example, an email service may require that a text editor be available.

Task ComputingMasuoka, Parsia and Labrou (2003) is claimed to let end-users accomplish complex tasks easily in environments rich with applications, devices, and services.

In Task Computing, the functionality of the user's computing device (from applications and OS), of the devices in the environment, and of the available e-services on the Internet is transformed into services. These services are described by semantic web technologies such as OWL and OWL-S. Tasks are seen as composed services (e.g., the "View on Project" + "My File" task is composed of the "View on Project" and "My File" services).

The system's user interface presents the available basic services to the user and allows him/her to either choose a basic service or compose a complex service using multiple basic services. While this approach minimises the complexity in the underlying middleware, it doesn't provide a good user experience since the user has to spend significant time and effort to understand the services to compose and achieve the desired task. For example, one may not aware of that executing [Local File + Bank (File)] will make the file available through the "Bank (File)" service in the conference room for other attendees to copy.

11.12 Natural Languages

This approach uses performative of the speech act such as "I'd like you (i.e., machine) to display, given a product, the products references, descriptions, and prices" Sabouret (2002) or verb-object phrases such as "play video", ""display slide-show", "find route from A to B", "send a message to a user", and "view photos on projector" **?** to represent high-level user's tasks. Although this approach allow end-users to easily to specify their tasks, the system must have an ability of understanding natural languages. This is a hard problem, or even impossible.

Interplay Messer et al. (2006) is a system that is claimed to allow users to use a simple pseudo-English interface to issues home tasks without having to consider where a particular content is located and how to achieve those tasks.

The minimal pseudo sentence representation for a task consists of a verb, a subject (content-type or content) and target device(s). For example, "Play (verb), The Matrix (subject), Living Room DTV (target device)" means "Play the DVD 'The Matrix' from the Bedroom DVD Player onto the DVT in the Living room".

11.13 ECO Language

explicit program-like representations: ECO Language Butler et al. (2007) is able to represent composite tasks composed of procedures, for example: "make coffee; turn lights dim; wait for lights; download news; wait for news; show news on tv;". An ECO procedure consists of *task statements* and *dependency statements*. A task statement is actually a particular command sent to a particular agent. A dependency statement specifies that a response from a certain task statement is a prerequisite for the execution of another task statement. the operator ";" indicates a parallel composition. In the above example, the term *coffee* refers to the brewing agent in the coffee machine. The coffee machine makes only a single style of coffee. If it could make multiple styles of coffee, the coffee style would need to be an argument, for example, "make Turkish coffee".

Clearly, this approach is an structural extension of the approach of using natural languages. So, it possesses the same problem of understanding natural languages.

11.14 Data-Action Pairs, verb-object phrases

A user requests a task by specifying a data file and action directive pairKumar et al. (2002). The extension of the file is used as the mime type of the request. For example, to request that the system play the movie “The Matrix”, the user would specify the directive “play” and the data source “TheMatrix.mpeg”.

verb-object phrasesMasuoka, Parsia and Labrou (2003)

11.15 TERESAXML

TERESAXMLMori et al. (2004) was the first XML language for task models where logical descriptions of requirements are translated into user interface description languages.

11.16 UIML

UIMLMir Farooq and Abrams (2001)

11.17 XIML

XIMLPuerta and Eisenstein (2002)

These languages, however, are not intended to render the user interface, from the task model, based on user needs or context but rather describe the structure of the interface and the interactions of the interface elements.

However, most of such approaches focus on providing device-adaptation support only in the design and authoring phase.

11.18 UMLi

UMLida Silva and Paton (2003)

11.19 Diane+

Diane+Tarby and Barthet (1996) Data Flow Diagrams may be used for the user interface task model.

11.20 JUST-UI

JUST-UIMolina et al. (2002)

11.21 SUPPLE

SUPPLEGajos and Weld (2004)

11.22 Teallach

TeallachGriffiths et al. (1999)

11.23 Wisdom

WisdomNunes and Cunha (2001)

11.24 Executable ConcurTaskTrees

Executable ConcurTaskTrees Klug and Kangasharju (2005) is an extension of task models using ConcurTaskTreesPaternò et al. (1997) (CTT) to allow dynamic execution of a task model. Firstly, task states and transitions between those states are appended to the CTT task model as shown in Figure 2. Secondly, input and output ports of particular tasks which enable information exchange between tasks are not directly connected.

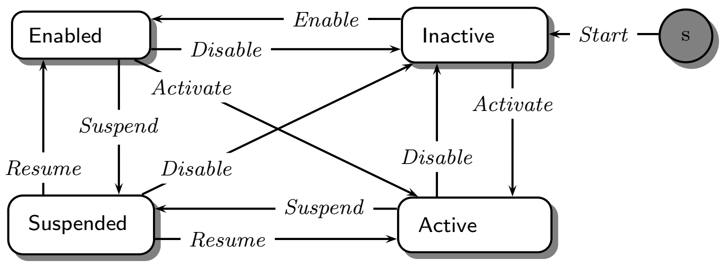


Figure 2: Task states and transitions in CTT

propose additional modelling constructs, namely input/output ports and object dependencies, respectively

11.25 Collaborative ConcurTaskTrees

Collaborative ConcurTaskTreesMori et al. (2002) support the specification of collaborative (multi-user) interactive systems. CCTT uses a role-based approach. A CCTT specification consists of multiple task trees. One task tree for each involved user role and one task tree that acts as a "coordinator" and specifies the collaboration and global interaction between involved user roles. Main shortcoming of CCTT is that the language does not provide means to model several actors, who simultaneously fulfill the same role.

11.26 Graph-Based Task Modelling

A Task Graph Pasu et al. (2001) (TG) describes a high-level task. It is a graph with nodes and edges. The nodes represent for abstract devices (or a group of devices) which offers a single service needed by the task while the edges represent for the associations between the nodes. A edge describes orders, properties, conditions, and data communicated among the nodes. A tuple architecture is used for data flows between the nodes.

Execution of a TG (a task) is actually a instantiation of the TG into a MANET. This work present two optimal algorithms (centralised one and distributed one) for instantiation a TG into a MANET. These algorithms allows the system rapidly adapt to the disruption due to the environment changes by dynamic establishment of the segment of the TG affected by this disruption. Context information used in the progress of instantiation is proximity and devices specifications.

The centralised algorithm requires that the user device (where the task is originated) has to execute the algorithm and has the complete knowledge of the network topology at runtime.

The distributed algorithm requires that every device in the network has to store this algorithm and has an ability to execute the algorithm when needed. Moreover, the user device (where the task is originated) needs to execute a specific algorithm. This demands every devices has a copy of this algorithm if it would like to be a user node.

The limitations of this approach:

- This work does not deal with sharing devices and conflict problem due to concurrent task execution sharing the same devices in the environment;
- It does not mention about pre-context and post-context while executing a task. Context information may be useful for the instantiation of a TG into a MANET at runtime;
- The approach does not capture user's feedbacks and user's intents which can be employed in order to provide the best support meeting user's requirements;
- It requires devices involved in a task having an ability for executing the instantiation algorithms and having the complete knowledge of the network topology at runtime. This further requires powerful computation capability of the device. This is opposed to the device constrains.

11.27 Business Process Execution Language

One of the notable languages for describing tasks as business processes is Business Process Execution Language[4] (BPEL or WS-BPEL). WS-BPEL is an XML-based programming language to describe interactions between business processes and Web Services. A business process is a collection of related, structured activities that achieve a particular goal. The activities of the processes are Web services. Because human and device interactions are not in its consideration, there is a significant gap for many real-world business processes.

In BPEL, the web services used and the pattern of interaction are pre-specified statically and hence the BPEL specifications are difficult to adapt to different situations.

[4]http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf

11.28 The Olympus Framework

In The Olympus Framework Ranganathan (2005), a task specification is written either in a C++ program or in a Lua Ierusalimschy et al. (1996) script that composes together different activities. Activities are written in the form of C++ functions using a high-level programming model called Olympus Ranganathan et al. (2005). Olympus consists of two elements called high-level operators and high-level operands. Examples of operators include starting, stopping and moving components, notifying end-users, and changing the state of various devices and applications. The operators are executed by using the specific libraries deployed for the space. Examples of operands are services, applications, devices, physical objects, locations, and users. Each type of operands corresponds to a class in the ontology that describes different classes, their relationships, and their properties.

11.29 PetriNet-Based Task Model, Semantic-based and Process-based Task Model

Pan et al. (2010) A Petri-net-based task model is proposed for modelling both task process and internal states, describing a task as coalition of primitive tasks. model a task from two aspects: task process and semantic description of primitive tasks. The task process describes the processing flow of the primitive tasks in a task, which is modelled based on Petri-net; the semantic description represents the semantic information of a primitive task, which is modelled based on Ontology.

Petri-net has powerful ability to represent a concurrent process both mathematically and conceptually. A Task Petri-Net (TPN) for the task process is represented by a 6-tuple:

$$TPN = \langle P, T, F, M, G, S \rangle, \text{where}$$

- P is a finite set of primitive tasks in the TPN. There is a special primitive task named "*End*" which once being activated represents the end of the task,
- T represents different stages of the task (called transitions),
- $F \subseteq (P \times T) \cup (T \times P)$ is the possible flow relation, which represents the relationship between primitive tasks and different stages of the task;
- M is a function that associates 0 or 1 with each primitive task in the net in realtime. The primitive task with 1 means it can be activated, namely, the primitive task can be carried out by an actual service immediately;
- G is a Guard function, mapping a Boolean expression to each stage of the task, which is used to judge whether the next primitive tasks can be activated or not;
- S is a function that associates primitive tasks to the *task-status set*, which represented status of the associated primitive tasks. The task-status set is specified as: {*Completed*, *Processing*, *Unstartted*, *Paused*}.

In a TPN, the initial status of all the primitive tasks is "*Unstartted*". When the guard function is satisfied, which means that the next primitive task(s) can be activated. If a primitive task is finished, its status will become "*Completed*".

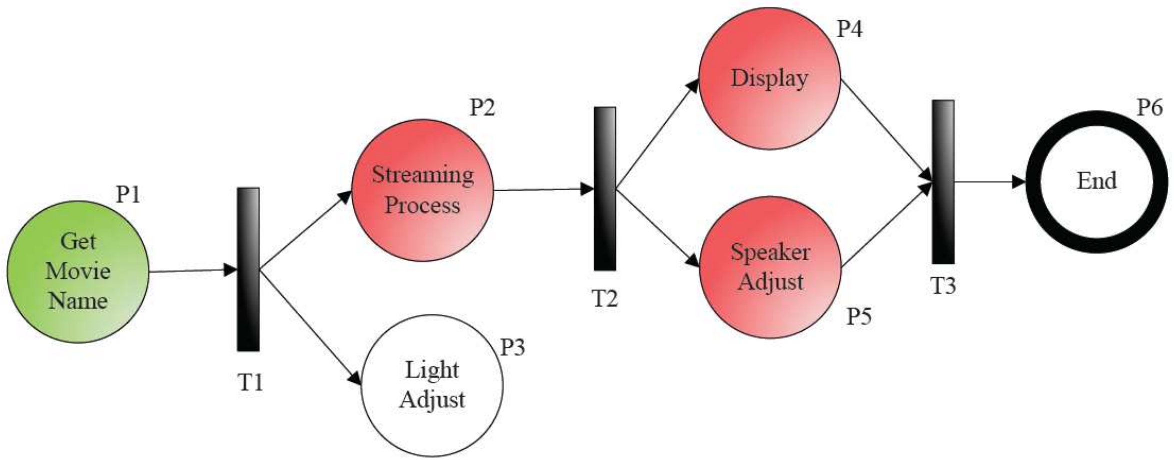


Figure 3: Task Petri-Net of the "Watch Movie" task.Pan et al. (2010)

With the TPN model, the primitive tasks' status can be tracked, which is useful to achieve task continuity when users move across smart environments.

In addition, a TPN model has the capability to describe if a primitive task in a TPN is optional or mandatory. A primitive task is optional in a TPN if it has no flow to the "*End*" task. Figure 3 shows the realtime TPN of the "watching movie" task. Assume that a user wants to migrate the task from car to home, green circles in the figure indicate the "*Completed*" status of primitive tasks, white circles indicate the "*Unstartted*" status of primitive tasks, and reds are "*Paused*".

11.30 OWL-S Process

In Amigo projectMokhtar et al. (2005), a user task is described as an abstract OWL-S process with no reference to existing services, and each environment's service is described as a semantic Web service with a conversation.

11.31 CTML

Collaborative task modelling language (CTML)Wurdel (n.d.) define a collaborative task model as a tuple consisting of:

A set of roles A role specifies a stereotypical user of the environment in the current domain (e.g. Chairman and Presenter in a conference session);

A set of collaborative task expressions (one for each role) A task expression provides a behavioural description of the assigned role. It may contain task dependencies to other task expression. Besides the hierarchical structure and temporal operators, a collaborative task expression consists of a set of tasks, each defined by an identifier, task type, a set of pre-conditions and effects. There are four types of preconditions:

Simple Task Precondition It enhances temporal operator by adding additional execution constraint within the collaborative task expression of the role;

Cooperative Task Precondition It addresses tasks of other roles defined in the CTML specifications. Because there may exist several actors fulfilling the same role, these preconditions need to be quantified using the *allInstance*, *noInstance* or *oneInstance* quantifier. They address multiple tasks, namely of each actor fulfilling the role.

Domain Preconditions They are used to restrict the task execution with regard to the domain model;

Location Preconditions They address the location of the actor performing the task.

A set of external models (such as the domain model or location model) They are used to specify dependencies to relevant entities whose state can be used to define task execution constraints (such as the domain or location model); and

A set of configurations A configuration specifies a runtime configuration of the CTML specification. It defines the set of actors including the assigned roles as well as the domain model instance (object model). Each actor belongs to one or more roles. A runtime configuration is the handle to animate the CTML specification.

Different types of relation between these entities exist: *depend*, *use*. A role may *depend* on tasks of another role which means that the task performance of the target role needs the execution of certain tasks from the source role. The *use* relation specifies that certain modelled objects are needed to accomplish the task set of the target role. Additionally, the referenced objects can be manipulated via task execution.

Task modelling is the process of developing a task model description Rich (2009).

A method of agent task representation and processing in pervasive computing environment

A Semantically-Based Task Model and Selection Mechanism in Ubiquitous Computing Environments

Ad hoc composition of user tasks in pervasive computing environments

Adaptable Pervasive FlowsAn Emerging Technology for Pervasive Adaptation

An EGA rule-based task programming language for ubiquitous environments

Context-Aware Active Task Discovery for Pervasive Computing

Context-Dependent task computing in pervasive environment

Flow-Driven Ambient Guidance

Making task modeling suitable for smart environments

Ontology-based high level task composition in ubiquitous computing applications

OWL-Based Context-Dependent Task Modelling and Deducing

Task Modelling Using Situation Calculus

Task Modelling for Ambient Intelligent Environments: Design Support for Situated Task Executions A goal-oriented interface to consumer electronics using planning and commonsense reasoning

A task-based interface to legal databases

A task-centered approach for user modelling in a hypermedia office documentation system

A Task-Driven User Interface Architecture for Ambient Intelligent Environments

A task-oriented interface to a digital library

Building task-based user interfaces with ANSI/CEA-2018

Design and evaluation of a task-based digital library for the academic community

Huddle: automatically generating interfaces for systems of multiple connected appliances

The Task-Based Interface: Not Your Father's Desktop

12 Conclusion

References

Annett, J. and Duncan, K. (1967). Task analysis and training design, *Occupational Psychology* **41**: 211–221.

Bastide, R. and Basnyat, S. (n.d.). Error patterns: Systematic investigation of deviations in task models, *Task Models and Diagrams for Users Interface Design* pp. 109–121.

Bolognesi, T. and Brinksma, E. (1987). Introduction to the ISO specification language LOTOS, *Comput. Netw. ISDN Syst.* **14**(1): 25–59.

Breiner, K., Görlich, D., Maschino, O., Meixner, G. and Zühlke, D. (2009). Run-time adaptation of a universal user interface for ambient intelligent production environments, *in* J. A. Jacko (ed.), *HCI (4)*, Vol. 5613 of *Lecture Notes in Computer Science*, Springer, pp. 663–672.

Butler, G., Loke, S. and Ling, S. (2007). Device ecology workflows with semantics: Formalizing automation scripts for ubiquitous computing, Online.
URL: *http://homepage.cs.latrobe.edu.au/sloke/papers/eco.pdf*

Card, S. K., Moran, T. P. and Newell, A. (1983). *The Psychology of Human-Computer Interaction*, Lawrence Erlbaum Associates, Hillsdale, NJ.

Coen, M. H. (1998). Design principles for intelligent environments, *AAAI '98/IAAI '98: Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, American Association for Artificial Intelligence, Menlo Park, CA, USA, pp. 547–554.

Cousins, S. B. (1996). A task-oriented interface to a digital library, *CHI '96: Conference companion on Human factors in computing systems*, ACM, New York, NY, USA, pp. 103–104.

da Silva, P. P. and Paton, N. W. (2003). User interface modeling in umli, *IEEE Softw.* **20**(4): 62–69.

Fischer, G. (2001). Articulating the task at hand and making information relevant to it, *Human-Computer Interaction* **16**(2): 243–256.

Fukazawa, Y., Naganuma, T., Fujii, K. and Kurakake, S. (2005). A framework for task retrieval in task-oriented service navigation system, *OTM Workshops 2005*, Springer Berlin/Heidelberg, pp. 876–885.

Fukazawa, Y., Naganuma, T., Fujii, K. and Kurakake, S. (2006). Service navigation system enhanced by place- and task-based categorization, *MobiSys 2006*.

Gajos, K. and Weld, D. (2004). Supple: automatically generating user interfaces, *IUI '04*, ACM, New York, NY, USA, pp. 93–100.

Garlan, D., Siewiorek, D., Smailagic, A. and Steenkiste, P. (2002). Project Aura: Toward distraction-free pervasive computing, *IEEE Pervasive Computing* **1**(2): 22–31.

Goth, G. (2009). The Task-Based Interface: Not Your Father's Desktop, *Software, IEEE* **26**(6): 88–91.

Griffiths, T., Paton, N. W., Goble, C. A., West, A., Barclay, P. J., Kennedy, J., Smyth, M., McKirdy, J., Gray, P. D. and Cooper, R. (1999). Teallach: A model-based user interface development environment for object databases, *Proceedings of the User Interfaces to Data Intensive Systems*, Washington, DC, USA.

Hartson, H. R. and Gray, P. D. (1992). Temporal aspects of tasks in the user action notation, *Hum.-Comput. Interact.* **7**(1): 1–45.

Heider, T. and Kirste, T. (2002). Supporting goal-based interaction with dynamic intelligent environments, *Proceedings of European Conference on Artificial Intelligence*, IOS Press, pp. 596–600.

Ierusalimschy, R., de Figueiredo, L. H. and Filho, W. C. (1996). Lua—an extensible extension language, *Softw. Pract. Exper.* **26**(6): 635–652.

Ikeda, M., Seta, K., Kakusho, O. and R., M. (1998). Task ontology: ontology for building conceptual problem solving models, *Proceedings of ECAI98 Workshop on Applications of ontologies and problem-solving model*, pp. 126–133.

Klug, T. and Kangasharju, J. (2005). Executable task models, *TAMODIA '05: Proceedings of the 4th international workshop on Task models and diagrams*, ACM, New York, NY, USA, pp. 119–122.

Kumar, R., Poladian, V., Greenberg, I., Messer, A. and Milojicic, D. (2002). User-centric appliance aggregation, *Technical report*, HP Labs.

Leung, Y. K. and Apperley, M. D. (1994). A review and taxonomy of distortion-oriented presentation techniques, *ACM Trans. Comput.-Hum. Interact.* **1**(2): 126–160.

Luther, M., Fukazawa, Y., Wagner, M. and Kurakake, S. (2008). Situational reasoning for task-oriented mobile service recommendation, *The Knowledge Engineering Review* **23**(1): 7–19.

Masuoka, R., Labrou, Y., Parsia, B. and Sirin, E. (2003). Ontology-enabled pervasive computing applications, *IEEE Intelligent Systems* **18**(5): 68–72.

Masuoka, R., Parsia, B. and Labrou, Y. (2003). Task computing–The semantic web meets pervasive computing, *Proceedings of the Second International Semantic Web Conference*, Lecture Notes in Computer Science, Springer Berlin Heidelberg, Florida, USA, pp. 866–881.

Meixner, G., Seissler, M. and Nahler, M. (2009). Udit - a graphical editor for task models, *Fourth International Workshop on Model Driven Development of Advanced User Interfaces (MDDAUI 2009)*, Florida, USA.

Messer, A., Kunjithapatham, A., Sheshagiri, M., Song, H., Kumar, P., Nguyen, P. and Yi, K. H. (2006). InterPlay: A middleware for seamless device integration and task orchestration in a networked home, *Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications*, IEEE Computer Society, pp. 296–307.

Meyyappan, N., Foo, S. and Chowdhury, G. (2004). Design and evaluation of a task-based digital library for the academic community, *Journal of Documentation* **60**(4): 449–475.

Mir Farooq, A. and Abrams, M. (2001). Simplifying construction of multi-platform user interfaces using uiml, *European Conference UIML*.

Mizoguchi, R., Vanwelkenhuysen, J. and Ikeda, M. (1995). Task ontology for reuse of problem solving knowledge, *Proc. of Knowledge Building & Knowledge Sharing*, pp. 46–59.

Mokhtar, S. B., Georgantas, N. and Issarny, V. (2005). Ad hoc composition of user tasks in pervasive computing environments, *Software Composition* pp. 31–46.

Molina, P., Melia, S. and Pastor, O. (2002). JUST-UI: A user interface specification model, *Proceedings of CADUI*, pp. 63–74.

Mori, G., Paternò, F. and Santoro, C. (2002). Ctte: support for developing and analyzing task models for interactive system design, *IEEE Trans. Softw. Eng.* **28**(8): 797–813.

Mori, G., Paternò, F. and Santoro, C. (2004). Design and development of multidevice user interfaces through multiple logical descriptions, *IEEE Trans. Softw. Eng.* **30**(8): 507–520.

Naganuma, T. and Kurakake, S. (2005). Task knowledge based retrieval for service relevant to mobile user's activity, *The Semantic Web – ISWC 2005*, pp. 959–973.

Naganuma, T., Luther, M., Wagner, M., Tomioka, A., Fujii, K., Fukazawa, Y. and Kurakake, S. (2006). Task-oriented mobile service recommendation enhanced by a situational reasoning engine, pp. 768–774.

Nichols, J. and Myers, B. A. (2009). Creating a lightweight user interface description language: An overview and analysis of the personal universal controller project, *ACM Trans. Comput.-Hum. Interact.* **16**(4): 1–37.

Nunes, N. J. and Cunha, Jo a. F. a. e. (2001). Wisdom: a UML based architecture for interactive systems, *DSV-IS'00: Proceedings of the 7th international conference on Design, specification, and verification of interactive systems*, Springer-Verlag, Berlin, Heidelberg, pp. 191–205.

Pan, G., Xu, Y., Wu, Z., Yang, L., Lin, M. and Li, S. (2010). Task follow-me: Towards seamless task migration across smart environments, *Intelligent Systems, IEEE* .

Pan, G., Xu, Y., Wu, Z., Yang, L., Lin, M. and Li, S. (2011). TaskShadow: Toward Seamless Task Migration across Smart Environments, *IEEE Intelligent Systems* **26**(3): 50–57.
URL: *http://doi.ieeecomputersociety.org/10.1109/MIS.2010.32*

Pasu, B., Ke, W. and Little, T. D. (2001). A novel approach for execution of distributed tasks on mobile ad hoc networks, *Technical report*, Boston University.

Paternò, F. (2001). *Handbook of Software Engineering & Knowledge Engineering*, World Scientific Publishing Co., chapter Task Models in Interactive Software Systems.

Paternò, F., Mancini, C. and Meniconi, S. (1997). Concurtasktrees: A diagrammatic notation for specifying task models, *INTERACT '97*, Chapman & Hall, Ltd., London, UK, pp. 362–369.

Payne, S. and Green, T. (1986). Task-actions grammars: A model of the mental representation of task languages, *Human-Computer Interaction* **2**: 93–133.

Puerta, A. and Eisenstein, J. (2002). XIML: A common representation for interaction data, *7th International Conference on Intelligent User Interfaces*, pp. 214–215.

Ranganathan, A. (2005). *A Task Execution Framework for Autonomic Ubiquitous Computing*, PhD thesis, University of Illinois at Urbana-Champaign.

Ranganathan, A. and Campbell, R. H. (2005). Supporting tasks in a programmable smart home, *In ICOST 2005: 3rd International Conference On Smart Homes and Health Telematic – From Smart Homes to Smart Care*, Magog, Canada, pp. 3–10.

Ranganathan, A., Chetan, S., Al-Muhtadi, J., R.H.Campbell and Mickunas, D. (2005). Olympus: A high-level programming model for pervasive computing environments, *Third IEEE International Conference on Pervasive Computing and Communications*, Kauai Island, Hawaii, pp. 7–16.

Rich, C. (2009). Building task-based user interfaces with ANSI/CEA-2018, *Computer* **42**(8): 20–27.

Sabouret, N. (2002). A model of requests about actions for active components in the semantic web, *Proc. STAIRS 2002*, pp. 11–20.

Sasajima, M., Kitamura, Y., Naganuma, T., Fujii, K., Kurakake, S. and Mizoguchi, R. (2007). Task ontology-based modeling framework for navigation of mobile internet services, *IMSA'07: IASTED European Conference on Proceedings of the IASTED European Conference*, ACTA Press, Anaheim, CA, USA, pp. 47–55.

Sasajima, M., Kitamura, Y., Naganuma, T., Kurakake, S. and Mizoguchi, R. (2006). Task ontology-based framework for modeling users' activities for mobile service navigation, *Proc. of Posters and Demos of ESWC 2006*, Budva, Montenegro, pp. 71–72.

Sinnig, D., Wurdel, M., Forbrig, P., Chalin, P. and Khendek, F. (n.d.). Practical extensions for task models, *Task Models and Diagrams for User Interface Design* pp. 42–55.

Tarby, J. and Barthet, M. (1996). The DIANE+ method, *Computer-Aided Design of User Interfaces*, Namur University Press, Namur, pp. 95–119.

Wang, Z. and Garlan, D. (2000). Task-driven computing, *Technical report*, School of Computer Science, Carnegie Mellon University.

Wurdel, M. (n.d.). Towards an Holistic Understanding of Tasks, Objects and Location in Collaborative Environments, *Human Centered Design* pp. 357–366.