

# Towards a Natural Interface to Adaptive Service Composition

Tony O'Donnell, Steffen Higel, Aoife Brady, Vincent Wade

3rd June 2003

{Tony.ODonnell, Steffen.Higel, Aoife.Brady, Vincent.Wade}@cs.tcd.ie

## Abstract

As computers become ubiquitous in our every day lives, providing a means for true user empowerment will become crucial. Techniques which allow dynamic reuse and composition of existing pieces of networked software (services) can provide a key solution to this problem. However the task of bridging the gap between user capability and the complexities of service composition provides a number of interesting research directions. This paper proposes an architecture for coupling an adaptive and semi-automated service composition system with a natural interface system, examined in the context of a university with a wireless network. The goal of this architecture is to assist in bringing a simplified but ultimately more powerful computing experience to the end user.

## 1 Introduction

Computer systems have developed such that users communicate with their machines in explicit terms and make use of static, persistent applications. This paradigm is by definition limited due to the constraints of that relationship. However, if one moves to a ubiquitous computing model, that relationship is fundamentally changed and so a new paradigm must be proposed.

In smart spaces, input and output can span a much wider range of devices, with an increased potential for natural interface technologies such as voice and gesture recognition. This liberates users from the conventional mouse\keyboard model, which is neither expressive nor intuitive, and allows them to instead interact through more natural means.

In addition, a monolithic, persistent application model becomes cumbersome and unhelpful in such a dynamic environment. A more promising alternative is to make use of ad-hoc service composition to facilitate the real-time composition of lightweight services to produce user-specific, disposable solutions.

Servicing users in such an environment presents two principle problem areas. The first challenge is to correctly infer a user's goals based on observation of their activity. Following on from that, a dynamic, ad-hoc composition of services must take place to satisfy this presumed need. This paper proposes a solution to both problems, which will see a symbiotic relationship between an *orchestration environment* charged with inferring a user's intentions and a *service composition engine* that will take requests and use them to design solutions in real-time. The two components operate in tandem to transparently satisfy user requests posed in a non-specific, natural way.

We will in particular examine how this combination can integrate into a learning environment, and how it can realise some existing theories in empowered learning.

## 2 Background

### 2.1 Service Composition

The field of service composition covers the techniques used to bind together two or more applications capable of intercommunication. Recent interest in the topic has surged due to the emergence of web-services - applications running on the World Wide Web, which use standardised methods for communication and self description[Edgar, 2001]. By chaining a number of these services together, functionality and convenience can be added to the services to which an end user might traditionally be accustomed - those offered by a single vendor accessed with a web browser.

The familiar example of buying a book online can be used to demonstrate this. A user would use a search engine to find a book about a given topic. She

reads reviews and decides on an appropriate title. She then goes to various online bookstores, though only the last one she checks has the book in stock. She orders the book, but because she doesn't have the only credit card the store accept, she must make the payment through some third party payment service.

Assuming all of the components that make up the selection and buying process offer web service interfaces to their business, someone with the skills to do so could provide a wrapper program which takes the output from each service, interprets it and passes it to the next. In essence, the wrapper program represents a service composition. The emergence of a number of technologies are starting to make the generated wrapper more sophisticated (to the point where 'wrapper' has become an inappropriate name for it) and the methods used to describe and locate appropriate services more advanced.

If we term the output of a service composition a compound service, it is also worth noting that any future service composition could include this compound service as part of itself. This recursive re-use of services can lead to faster compositions and more efficient service location[Yang, Papazoglou, 2002].

### 2.1.1 WSDL

WSDL is a language for describing services in terms of the operations they perform on messages (the 'how' of web-services, as opposed to DAML-S' 'what'). The service is essentially defined as a number of required inputs and a number of expected outputs. It can describe the supported protocols of a service, the format of the messages it can receive and can facilitate the binding of the service to lower level protocols like HTTP and SOAP.

It becomes useful when we are supplied with two services, and would like to facilitate communication between them. If the service creators supply a WSDL description of their services, it would become relatively simple to derive their inputs and outputs and then provide a custom mapping from one to the other. For example, we could have one service which accepts credit card payment and another which sells a product to an end user. The WSDL description of the credit card service could specify a function called Accept-Details, which accepts CreditCardNumber (integer) and ExpiryDate (string) as inputs, and PaymentAuth (integer) as an output. The service which sells the product accepts AuthorizationCode (string) as an input. Armed with this knowledge, a quick transformation of the data is all that is required to get the two communicating properly.

### 2.1.2 DAML-S

The DARPA Agent Markup Language (DAML) is an RDF based language used to describe information in manner that allows software to interpret it meaningfully. On the other hand, the most widely used markup language, HTML, is used to allow a web browser to format and display the information in a user interpretable manner. DAML in particular is interested in describing the relationships between different pieces of information (or ontologies), and providing a form of context to the information itself. Perhaps a good example of its usefulness is in dealing with words which have multiple meanings. The word spirit is one such word, taking on different meanings and implications depending on the context in which it is used. The human mind can usually extract some notion of what context is being implied. Computer programmers do not have this luxury when developing software. This is one major issue tackled by DAML.

DAML-S is an extension built on top of DAML, much like DAML itself is built upon RDF, which provides an ontology for web services. The project's stated goal is to create a computer interpretable markup language for describing the properties and capabilities of web services, facilitating the automation of Web service discovery, execution, inter operation, composition and execution monitoring. DAML-S describes a service in terms of three major areas:

*Service Profile* - This covers the preconditions, inputs, outputs and effects of the service.

*Service Model* - This describes how the service works, facilitating the automated invocation, inter-operation and monitoring of the service.

*Service Grounding* - This defines the service access information specification; the communications protocols and transport mechanisms etc.

As these technologies such as DAML-S and WSDL (and those that underly them) mature, they will continue to facilitate the application of methods from other areas of knowledge based computing to service composition [conlan03a].

## 2.2 Natural & Disappearing Interfaces

In the beginning, many people used a single computer. Then came the desktop PC and the relationship shifted to that of a single user with their own machine. Now with the ever-decreasing size and increasing power of comput-

ers, it is possible to embed processors all around us, and allow a single user simultaneous access to many computing devices. As a result the notion of a computer as a distinct machine will be replaced with a ubiquitous ambient computing presence[Dey et al, 2001].

Of course, such a radical shift in computer use will necessitate a re-examination of the mode of interaction between users and their machines. It may no longer be acceptable to offer specific interfaces to individual components and so the interface must itself become universal and transparent[Horvitz et al, 2003]. While this presents a daunting challenge to interface designers, it also offers liberation for end-users, especially if natural interfaces are embraced.

Natural interfaces shift the debate on human-computer interaction away from making interfaces that suit the formality of a computer, and instead focus on using techniques that match how humans interact with each other, e.g. a mouse and keyboard may provide inputs that the computer can easily understand, but a person would much rather request services verbally. They allow interaction through conventional inter-personal techniques such as speech and gesture, and therefore require no special training for people wishing to access the system.

This could prove profoundly empowering for users. Human beings are not naturally inclined towards the rigidity of the keyboard\mouse model, nor are we particularly adept at processing precise, logical information. Eons of evolution have, however, made us adept at manipulating our physical environment and dealing with imprecise information[Wang et al, 1997]. The goal in smart spaces will be to implement such natural interfaces so that interaction with the system will be as easy as waving a hand or uttering a request in natural language.

### **2.3 Adaptive Techniques**

Software in its traditional form has been driven predominantly by a fixed set of design decisions, defined before development begins, which place finite limits on the capabilities of the end product. Although time has been spent on research into software which can learn the habits and preferences of the user[Langeley 1997, Gervasio et al, 1998], it is only in recent years that we have seen any definite move in the direction of products being used by end-users which feature adaptive components[Horvitz, 1999]. These agents can monitor a users interaction with software and take actions on behalf of the user based on these assumptions.

The emergence of the World Wide Web as a method for sharing information and providing business services also spawned an evolutionary branch of research which has converged on methods and thinking similar to those of adaptive software. High-level languages designed specifically for web development like PHP and ASP have enabled web developers to provide end-users with the ability to configure attributes of web-sites according to their tastes. This then progressed into technologies like AvantGo which translated the contents of news websites into a format readable on a PDA.

From this, two adaptable components of human-computer interaction can be seen; navigation and presentation. If we view the user's interaction with a piece of software as a passage through a number of required sub-tasks (write the document, format the document, print the document etc.), alternative ways of interacting with and configuring these subtasks could be provided. Thus, the path that the user takes through these alternatives (or candidates) could be adapted to their needs, assuming we have a suitable description of both the user's capabilities and preferences and the candidate sub-task provider's operational methods. The process of selecting and presenting a series of candidates to a user is termed adaptive navigation.

Adaptive presentation covers much of what was described previously. The challenge remains in defining a way of adapting the look and feel of software or information in a generalised manner.

### 2.3.1 Candidacy

In adaptive hypermedia systems, although *metadata* (the data which describes the data) driven techniques made some headway in describing the scope of a piece of content, it did not operate on a sufficiently high level to give any meaningful hints to the system itself as to what goals this piece of content was trying to fulfill[Dagger et al, 2003]. The navigation through a collection of hypermedia documents is abstracted such that it can be viewed as a navigation through a series of concepts represented as nodes. These concepts will have one or more pieces of appropriate content associated with them.

Any node which has multiple suitable pieces of candidate content will require a decision to be made by the adaptive system. One of these candidate pieces of content will eventually be selected. It should be noted that all of the available candidates will provide the same end effect, though the presentation or means of delivery will differ. This idea of *candidacy* can effectively be

applied to another areas of adaptive software, as will be demonstrated in this paper.

## **2.4 Ubiquitous Computing in Education**

Ubiquitous computing will have many applications, and education is sure to be among them. The integration of the World Wide Web into classrooms has shown how new technologies can enable fresh approaches to learner-driven education, and ubiquitous computing has the power to further empower learners. This paper will examine how this empowerment can be delivered, especially how ubiquitous computing can facilitate portfolio-based learning and adaptive content delivery.

### **2.4.1 Portfolio-Based Learning**

The notion of portfolio-based learning has been around for 20 years, and is now becoming a preferred technique in institutions in the US and UK such as the University of Westminster, University of Newcastle and the University of Texas. A portfolio contains a record of all graded work done by a learner, as well as any relevant professional experience [Brown, 2001]. Unlike an examination transcript, a portfolio offers more depth and allows a viewer to glean a more complete picture of the learner's acquired skills. In addition, since it also contains relevant experience, the portfolio can assist life-long learning, accompanying the learner as they move through industry and academia.

Portfolio-based learning also makes heavy use of Project-based learning (PBL), in keeping with the Aalborg Model. PBL is an instructional method that challenges students to work cooperatively in groups to solve problems. A smart space environment could prove an invaluable support to this type of activity, as it could assist with scheduling, version control, minuting, etc.

### **2.4.2 Adaptive Content Delivery**

Adaptive techniques allow tailored course delivery so that learners receive content tuned to their personal learning style and skill-set [Brusilovsky, Eklund, 1998]. This adaptation can either affect navigation or presentation, such that different users can cover the same material but in different ways. The arrival of the World Wide Web enabled a major leap forward in this field through the use of adaptive hypermedia, and there can be little doubt that adaptive

content will be further assisted by ubiquitous computing, especially given the potential proliferation of display and delivery devices.

## 3 Research Directions

### 3.1 Proposed Methods of Interaction

#### 3.1.1 Requirements

The above advances in research have the potential to empower technologists and end-users alike. However it is the belief of the authors that for these ideas to become truly accessible and convenient to users, a meaningful interaction must take place between them. These concerns motivate the discussion and ideas in the following section.

Any model of these interactions should be built around a balance of the following requirements:

**Orthogonality** The conceptual model of each component should be such that no functionality or roles are reproduced. Furthermore, an appropriate location for any additional functionality or role should be readily apparent. This helps to define cleanly the breakdown of research focus.

**Efficiency** A well designed model should take into consideration the flow of information between components. Frequently required information should be made as available as possible to any core component.

**Usability** It is obvious that if the purpose of the model is to provide a user-driven interface to complex notions, the model must define where the users' interactions end, and exactly how much is required of them.

It is clear from; previous discussion that there will be two core components in our solution defining the interaction so as to provide a more powerful and usable experience to the user: the orchestration environment (responsible for interpreting the users actions based on sensor data and this abstracted concept of the user) and the service composition engine (responsible for taking the orchestration environment's interpretations of the user's needs and converting them into compound services).

At the core of this interaction is the information passed from the orchestration environment to the service composition engine. The format and scope of this data will draw clear lines between the responsibilities and requirements of each component. This data will describe a task that the TSUNAMI has interpreted from the actions of a user.

The problem is finding a good balance between an extremely detailed description of a task (which would expect too much of the TSUNAMI and too little of the SCE) and an extremely vague description of a task. Perhaps examining the functionality of the two systems will provide a useful hint. The TSUNAMI interprets a user's actions and attempts to take some meaningful information from them. The SCE is required to take a given task description, break it down into a number of suitably granular subtasks and then resolve those into given services. It would seem reasonable to assume from this that the service composition engine must be fed some structured description of the task, with much of the contextual interpretation already performed. It is after all, the orchestration environment which has this information readily available to it.

It is therefore proposed that the task description passed from TSUNAMI to SCE be expressed in terms of conditions, effects and outputs, rather than a verbose specification. The generation and interpretation of those conditions, effects and outputs are still complex, but they place reasonable and logical requirements on each component. Using the buying-a-book example, conditions could be *user has sufficient funds in their bank account* and *book must exist*. Effects could include *user will be debited the cost of the book* and *book is delivered to user*. Outputs would include *generate a receipt*.

## 3.2 Proposed Model for Adaptive Service Composition

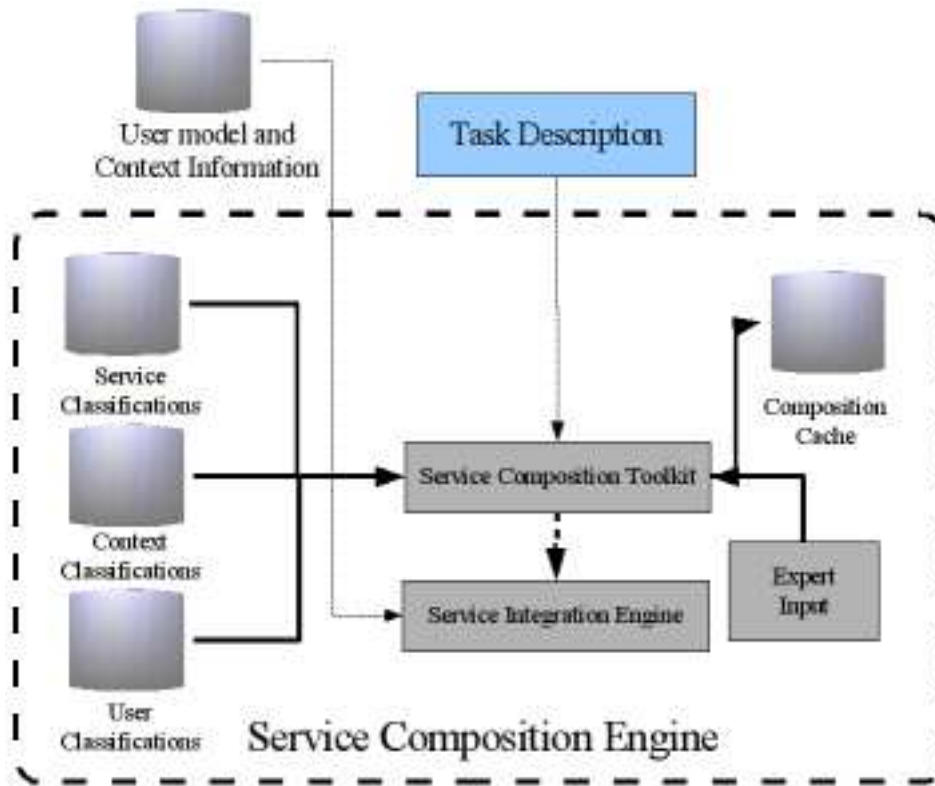
We propose an architecture for a service composition system which integrates the composition of services by a domain expert and the adaptive integration of services in response to user needs. This approach to service composition has the potential to provide a series of benefits to software developers and end users alike. From the developer's perspective, the speedy and semi-automated selection and linking of distributed components saves time and effort. From the end user's perspective, this approach could result in applications which are coupled more closely with their needs and preferences, along with providing a piece of software which offers levels of functionality greater than the sum of its parts.

### 3.2.1 Architectural Requirements

Taking this basic idea, an architecture should be formed around an extrapolation of the base requirements. If the overall objective of an adaptive service composition is to tailor the the composition to user needs, it should first and foremost be designed to be flexible in its required inputs. The description of users accepted by the system should not be hard-coded, rather, they should be dynamically mapped onto appropriate end points. If necessary, emerging techniques in semantic interoperability can be employed. If the system is to compose services which exist on large, dynamic networks (like the Internet), it should be capable of compensating for services which become unavailable. Finally, there is the goal of maximizing the re-usability of the service compositions. If each generated composition is seen as having a certain amount of stored intelligence within itself (and as such has some *value*), the system should store the compositions for future re-use. Tackling how best to express their worth will have a considerable influence on the architecture.

The service composition system is given some description of the overall task that is required by the end user and has sufficient logic in place to break the task own to atomic functionality provided by given services. To maximise the re-usability of the composition, it would seem logical to abstract a description of the steps required to fulfill the task from the actual description of how information should flow between the actual services. It is obvious that any number of factors could render a service unavailable or merely inappropriate for use. For this reason alone, it would be better to describe a node in a service chain in terms of its functionality, rather than providing the location of some actual service.

As a consequence of this design decision, the system should separate the process of decomposition of a given task to a series of subtasks and that of resolving those subtasks to actual services. This two phase process will aid in the goals of providing a system which promotes reuse, is capable of withstanding service failure and is sufficiently flexible to deal with differing user needs.



Despite the research aimed at advancing the field of semantic interoperability, there must be some fall-back mechanism for the engine to rely on, something with a sufficient level of skill to decompose a given requirement into subtasks. Capable of understanding the context in which a requirement is given and being far more adaptable to and tolerant of small variations in input, there is (at this point in time) no substitute for trained human input. It is therefore proposed that what we shall term an *expert* can provide the insight that a computer cannot acquire when attempting to decompose a service. As previously mentioned, any intelligent input into the system has some value, and as such should be stored in such a way as to maximise re-usability. In this case, a caching mechanism to buffer the supplied knowledge would be appropriate.

This adaptive service composition process should begin with the Service Composition Toolkit (SCT) being fed the general task requirements ultimately supplied by the end-user. A domain expert has some understanding of how a task can be broken down into smaller sub-tasks which could themselves map onto a composition of different service type groups (i.e. a group of

different services which provide similar functionality) likely to be available. The SCT will generate an abstract service composition that can fulfill the proposed user task and store this in a candidate repository made available to the Service Integration Engine (SIE). The SCT also uses input from a repository of User, Service and Context Classifications, which are used to devise preconditions and hints for the SIE in resolving a task to an appropriate candidate service. To improve task-to-composition mapping over time, the repository of previously developed services can be analyzed to see if a close match already exists.

### **3.3 Tailored Support of Users' Natural Activities with Mixed Initiative**

The Tailored Support of Users' Natural Activities with Mixed Initiative, TSUNAMI, is the link between the smart space's management technologies, the physical environment manipulated by the real world user and any other data which an attentive system might need to assist a user. It provides a well-defined interface through which a user, or the system monitoring the user, can interact with the TSUNAMI. It is capable of interpreting a user's vague requests and producing a set of requirements that an adaptive engine can then use to satisfy the user's needs. The TSUNAMI should maintain a list of available services, based on location and available resources, which can be used in conjunction with a user's privileges to generate a menu of available services that the adaptive engine can use to compose solutions. The TSUNAMI should be able to cache successful solutions from the adaptive engine so that in future a user can be offered a composition that previously worked, without burdening the adaptive engine. Finally, the TSUNAMI should have a feedback mechanism so that it can learn from mistakes, and allow the user to feel some degree of control.

#### **3.3.1 User Abstraction Layer**

The TSUNAMI is going to have to interact with a variety of different technologies and systems. This should be done transparently through a well-defined interface. This would make the system more versatile and generic. The interface should contain support for context managers, user-monitoring systems, and service directories, and also provide for the interaction of the TSUNAMI with the adaptive engine.

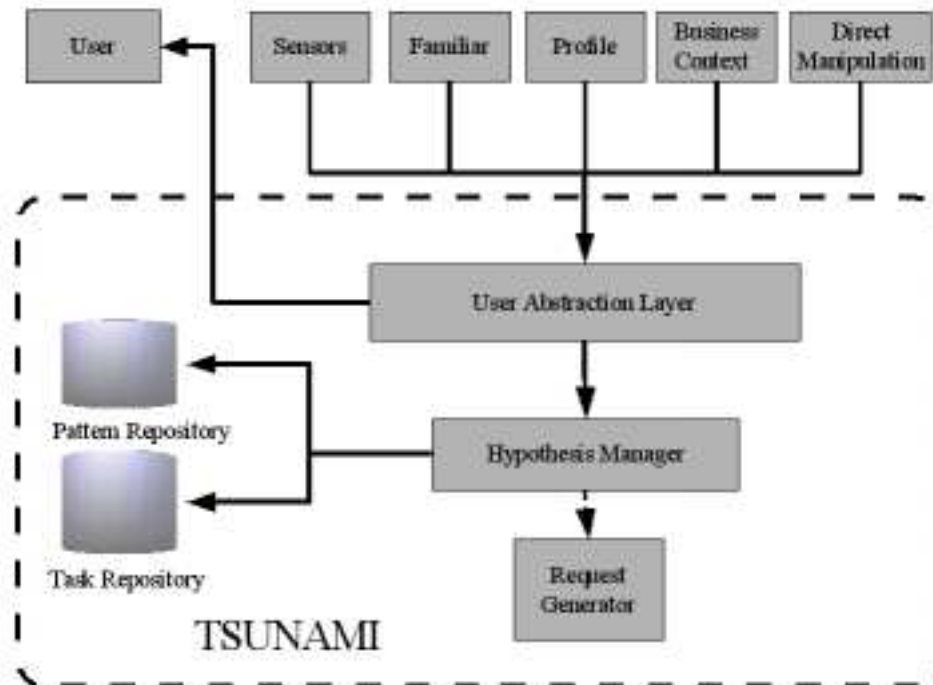
### 3.3.2 Interpreting User Activity

Human beings are instinctively good at physical manipulation of objects and dealing with imprecise information, while being precise and adept at dealing with abstract logic is not natural. The TSUNAMI should be able to use whatever monitoring resources are available in the smart space, e.g. cameras, microphones, IR receivers for smart badges, etc., to establish the user's identity and goals. The feed from the monitoring devices will be, by definition, largely imprecise and it will be the TSUNAMI's duty to process these raw observations and turn them into requirement sets that can be passed on to the adaptive engine in a standardised, well-formed format.

This constant stream of input data gives the TSUNAMI an advantage over traditional modes of adaptivity. In most adaptive systems, adaptation is driven by a user model. This can be a powerful tool, however it does have limitations, especially since it rarely, if ever, considers a user's intentions at a given moment. For example, if a news website adapted itself to a user's tastes, it may prioritise sports stories for a football fan, but that does not mean that the user only ever wants to see match reports and so at any given moment their intention might help to inform the adaptive process.

The TSUNAMI should therefore try to consider the user's intention context when it interacts with the adaptive engine. Intent can be inferred from a range of data including sensors observing non-specific behaviour, calendars, a user's familiar (an electronic companion storing user preferences and historical data), the conventional user profile, business process and the current context as well as through declarative tools within the space, such as traditional, explicit input devices.

In addition, the usefulness of a given derived intention changes depending on whether a task is to be performed unilaterally by a given user, or whether they are collaborating with others to achieve a common goal [Sullivan et al, 1999]. This presents a range of new factors including the social consequences of adopting a given action at the expense of others. The TSUNAMI should also resolve these potential conflicts before triggering a request to the Service Composition Engine.



### 3.3.3 Triggering Requests

As a result of the volume of input data the TSUNAMI receives, it will necessarily develop more than one hypothesis regarding the user's goals. In order to reconcile this, the system can make use of certainty thresholds to manage the triggering of requests to the adaptive engine [Horvitz, 1999]. Such thresholds operate in two ways.

Firstly, the more explicit a user's input, the higher the confidence the system can have in its prediction. For example, a typed-in command such as *print file* is very explicit and precise, while a comment on the heat in the room might not be a request to adjust the temperature. In the latter case, the system may have to make further observations in order to increase confidence. Secondly, the task at hand will set the level of confidence required. If the presumed task is mission-critical, then a very high degree of certainty will be required to trigger a request, while more trivial actions might not require the same certainty. The TSUNAMI should be able to perform arbitration between the two, and where necessary it may even prompt the user directly to confirm their presumed request.

### 3.3.4 Service Availability

A smart space can offer a user only those services which have been declared to it, and also those which can actually be delivered given the resources available, e.g. a map cannot be displayed when there are no free display or printing devices. The TSUNAMI may need to be aware of the available services and also the user's access privileges, e.g. have they paid for the service, so that the composition engine knows what services are at its disposal.

Solution\Requirement Set Caching Caching is useful for two reasons. Firstly, as in any system, caching can speed things up. If a user asks for a map of the surrounding area, then the solution generated today will be equally valid tomorrow, so the TSUNAMI can present the cached solution without resorting to the adaptive engine.

Caching may also be useful for quality control. Particularly successful requirements sets could become commercially valuable, while those generated in error can be useful for correcting bugs in the system and allowing it to learn from its mistakes.

### 3.3.5 Feedback

User empowerment is a key feature of any successful smart space project, and feedback can be used to satisfy this. This feedback can be explicit such as prompting a user for their comments; this would be especially useful if the TSUNAMI was dealing with a new user about whom it knew little, or implicit, such as listening for verbal complaints. The feedback could also be accomplished by interacting with the user's familiar.

## 4 Useful Application of Research Directions to Scenario

The goal of this paper is to define useful techniques for the interaction between roaming user models, natural interfaces and adaptive service composition. This section will tie these into a workable framework and demonstrate their potential benefits in the field of ubiquitous computing aided learning.

The scenario will show how our proposed system can aid problem-based learning and group work. We will demonstrate how our approach can assist with project planning, execution and submission, including how it can

interact with a portfolio management system.

Third year Java Lecturer, Dr. Glen Ryan has divided his class into project groups and set them an assignment to design a distributed file server. Project member, Declan Power, asks his diary manager to find a meeting time that suits the group. The TSUNAMI takes this request and converts it from a user level goal, to a formal task specification that can be passed to the SCE. Included in such a specification will be any pre- and post-conditions, as well as any constraints that could effect the composition offered. In this case, where the user level goal is to find a mutually acceptable time for a meeting, the task sent to the SCE might break down as follows.

### **Pre-conditions**

- user is Declan Power
- system must have access to student diaries
- Declan must have some access privileges to diaries other than his own
- diaries must be searchable

### **Post-conditions**

- diary entries can be changed
- diary owners can be notified of changes

### **Constraints**

- Declan has access to a standard display

The SCE understands that it is to broker the task of negotiating a meeting between the members. It resolves the task down into subtasks, namely accessing the diary of each group member, finding an appropriate time to hold the meeting, booking a room and finally ensuring that any materials necessary are present in the meeting room. These tasks are then resolved to services which the department provides, using the Service Integration Engine.

The services themselves have both WSDL and DAML-S descriptions available. The service that resolves an appropriate meeting time has its

preconditions and other effects defined using DAML-S. The user of this service must have received permission to access the other users' diaries before the service executes the request, and this is suitably defined in the DAML-S description. Likewise, the inputs and outputs of each function that each service provides are defined by WSDL. This aids the service composition system in automating the interaction between the different services. There is no need for the developers of the individual services to take other services into account.

Privacy being a core concern of the system, it is not simply possible for any given service to access a user's diary. This information is offered by some trusted representation of the user on the network to the service required to broker a suitable meeting time. Having successfully found one, a screen within Declan's field of view shows him a list of suitable times and he says "the second one" out loud and suggests they meet at the campus coffee shop. The TSUNAMI understands *the second one* and sends a request to the SCE to arrange a meeting at 2pm next Wednesday at the coffee shop. The SCE creates a service composition to book the meeting and send notifications to the relevant students. Once confirmation is received from everyone, the composed service sends a message to Declan's PDA. Declan's PDA beeps and tells him that the meeting has been arranged.

On the day of the meeting, Declan leaves his smart space enabled dorm and heads to the coffee shop. The TSUNAMI knows that Declan is going to a meeting and that the coffee shop is not networked. It contacts Declan's PDA and asks it to record the meeting and to upload it at the next opportunity. The group meet and discuss the approach they are going to take, the division of work and the common IDE they are going to use. The meeting concludes and Declan heads back to his room. The PDA has recorded the conversation and once Declan gets home it uploads it to the TSUNAMI. The TSUNAMI, provided with some contextual information relating to the recording it is given, scans the conversation for actions that it can help with, and it then sends a number of requests to the SCE including requests that it install Sun One on Declan's PC, and that it checks if Declan is trained in RMI. The SCE firstly takes on the task of installing Sun One. This requires a number of subtasks, namely obtaining the software package and then installing it to Declan's machine. Having checked its cache, the SCE finds that it has done such installations before and retrieves the service composition and applies the services to the current installation. It then moves on to the training issue. This task requires checking Declan's portfolio to see if he has passed

suitable modules. The system passes this data back to the TSUNAMI and asks for further instruction. The TSUNAMI receives Declan's record and realises that Declan has not used RMI in over a year. It polls his profile, which says that if a skill is stale, Declan likes to get a refresher. It then requests that a course is prepared for Declan's learning style and experience. The SCE prepares an on-line tutor using the university's content repository delivery service, and an adaptive tutor, which can tailor output to Declan's style. Declan logs into his PC and finds that Sun One has been installed and that there is a message in his Inbox linking him to an RMI refresher. As the deadline approaches the group meets again in the coffee shop to compare work. Their code has been synchronised by the smart space so that they all have the most up-to-date version. Declan is selected by the group to submit the work.

Declan returns to his room, and asks the system verbally to submit the work to Dr. Ryan. The TSUNAMI asks the SCE to submit the completed project. The SCE contacts the portfolio management service and receives a connection, which it passes to the TSUNAMI, along with a connection to the file service storing the work. The TSUNAMI retrieves the project and submits the work. It then asks the SCE to send a confirmation. The SCE uses an e-mail service to contact the group. Declan checks his mail and sees that his work has been sent in.

The Portfolio Manager can then take over and inform Dr Ryan that the students' work has been submitted. Once he has marked the work, the PM will send notices to the group with their marks.

## 5 Future Work

Although the core design and interaction of the system has been defined, extensive simulation is required to ensure that the stated goals are being met optimally. A testbed of sample services are needed before any work on forming adaptive compositions can begin. Preliminary work has begun in this direction, taking some of the more useful atomic programs from UNIX systems (notably the ps2pdf suite of commands, the file command used to identify files and various file retrieval commands like curl and wget) and providing SOAP wrappers for them, thus exposing them as services. Most of these commands are used regularly in lightweight shell scripts. Because there is a small degree of synergy between the ideas of shell scripting and

service composition (atomic, independent programs accept a stream of data, manipulate it and then pass it to the next), it will be of interest to observe how techniques used in scripting (which people have been doing since the early 1970s) can be applied to the younger field of service composition.

In parallel to this research, frameworks must be defined to enable the deployment and execution of composed services. Answers to questions such as from where should the execution of a composed service be actually controlled, how will assurance levels for composed services function and how can composed services interact with existing security mechanisms will only be answered through the development and use of a testbed.

Conceptual work on the orchestration environment can be divided into two areas. The first challenge will be developing models that will allow the system to understand events and tasks, events being inputs from the real world and tasks being perceived user goals. The other will centre on processing these models so that inputs can be matched against tasks allowing the TSUNAMI to generate requests to the SCE.

In order to assist with the preparation of models, we propose to draw on existing knowledge in the area of modelling such as the work of Eric Horvitz. Models will need to precisely and unambiguously capture data. They will also have to facilitate aggregation. This presents the further issue of semantic interoperability[?]. Models will also have to represent context-dependant confidence levels.

Processing these models in order to resolve inputs into reliable task requests will require building hypothesis maps made up of observed inputs which can then be matched against predefined goals. Arbitration between and selection across these maps will be a significant research focus. We would propose drawing on areas such as Complex Even Processing[Luckham, 2002] to facilitate these operations.

System testing will initially take place in a controlled environment, either simulated or physical, using simple models of behaviour and a limited set of available goals. Based on empirical evidence from this, more complex configurations could then be considered.

## 6 Conclusions

This paper has proposed techniques for bridging the gap between an end user and a service composition system using natural interfaces. The components

of this system were discussed, and their internal architectures were derived and justified. The system's uses were explored through the example of a mobile learning environment, with an emphasis on a tailored and user driven computing experience.

At this point, research should focus on bringing ideas from other areas of adaptive computing into those of service composition and natural interfaces. Existing research into adaptive hypermedia systems can provide solutions to some problems in this domain. Furthermore, testbeds should be developed to determine how flexible these systems are, allowing the concepts introduced here to be refined. Also important and worthy of future examination are the techniques used to deploy, execute and monitor services.

## References

- [Brown, 2001] The Portfolio: A Reflective Bridge Connecting the Learner, Higher Education, and the Workplace, Brown, Judith, *The Journal of Continuing Higher Education*, Vol. 49, No.2, Spring 2001
- [Edgar, 2001] Web Services: A Position Paper, Edgar, G, W3C workshop on Web services, 2001
- [Yang, Papazoglou, 2002] Web Components: A Substrate for Web Service Reuse and Composition, Yang J. and Papazoglou M. P., 14th International Conference on Advanced Information Systems Engineering, 2002
- [Brusilovsky et al, 1996] ELM-ART: An Intelligent Tutoring System on the World Wide Web, Brusilovsky, Peter, Eklund, John & Schwarz, Elmar, *Intelligent Tutoring Systems*, Lecture Notes in Computer Science No.1086
- [Brusilovsky, Eklund, 1998] The Value of Adaptivity in Hypermedia Learning Environments: A Short Review of Empirical Evidence, Brusilovsky, Peter & Eklund, John, *Proceedings of the 2nd Workshop on Adaptive Hypertext and Hypermedia*, June 1998

- [Dey et al, 2001] Distributed and Disappearing User Interfaces in Ubiquitous Computing, Dey, Anind, Ljungstrand, Peter, Schmidt, Albrecht
- [Ford, Bradshaw, 1993] Knowledge Acquisition as Modelling, Ford, Kenneth & Bradshaw, Jeffrey, International Journal of Intelligent Systems, Vol. 8, No. 1, January 1993
- [Horvitz et al, 2003] Models of Attention in Computing and Communication: From Principles to Actions, Horvitz, Eric, Kadie, Carl, Paek, Tim & Hovel, David, Communications of the ACM, Vol. 46, No. 3, March 2003, pages 52 - 59
- [Horvitz, 1999] - Principles of Mixed-Initiative User Interfaces, Horvitz, Eric, Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit, pages 159-166, 1999
- [Sullivan et al, 1999] - Intention Reconciliation in the Context of Teamwork: An Initial Empirical Investigation, David G. Sullivan<sup>1</sup>, Alyssa Glass<sup>1</sup>, Barbara J. Grosz<sup>1</sup>, and Sarit Kraus, Cooperative Information Agents III, Lecture Notes in Artificial Intelligence, Vol. 1652, Springer-Verlag, 1999, pp. 149-162.
- [Wang et al, 1997] - Object Manipulation in Virtual Environments: Human Bias, Consistency and Individual Differences Wang, Yanqing, MacKenzie, Christine L. & Summers, Valerie A., CHI 97 Electronic Publications, 1997
- [Dagger et al, 2003] An Architecture for Candidacy in Adaptive eLearning Systems to Facilitate the Reuse of Learning Resources, Dagger D., Conlan O., Wade V., Awaiting Acceptance
- [Luckham, 2002] The Power of Events, Luckham, David, Addison-Wesley, 2002

- [Langley 1997] Machine learning for adaptive user interfaces, Langley, P. Proceedings of the 21st German Annual Conference on Artificial Intelligence, 1997
- [Gervasio et al, 1998] Learning to predict user operations for adaptive scheduling, Gervasio, M., Iba, W., & Langley, P, Proceedings of the Fifteenth National Conference on Artificial Intelligence, 1998