THE FLORIDA STATE UNIVERSITY
COLLEGE OF ARTS AND SCIENCES

A Modular Data Pipelining Architecture (MDPA) for enabling Universal Accessibility in P2P Grids

by

Sangmi Lee

A Dissertation submitted to the
Department of Computer Science
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Degree Awarded:
Summer Semester, 2003

The members of the Committee approve the dissertation of Sangmi Lee defended on July, 3, 2003.

Gordon Erlebacher
Professor Directing Dissertation

Larry Dennis
Outside Committee Member

Geoffrey C. Fox
Committee Member

Gregory Riccardi
Committee Member

Robert A. van Engelen
Committee Member

The Office of Graduation Studies has verified and approved the above named committee members.

To my parents

ACKNOWLEDGMENTS

I would like to thank all those that have offered me their support and encouragement while working on this thesis.

In particular I would like to acknowledge the inspiration and guidance given to me by Professor Geoffrey Fox, supervisor of my research during my graduate studies. He gave me a lot of opportunities to work on great research issues and inspired me with his brilliant ideas. As time goes by, my appreciation for him will be deeper and deeper. I would like to thank my advisor Professor Gordon Erlebacher for his support me during my Ph.D.

I strongly appreciate the support and patience of my father, Kwang-Ho, who encouraged me in every step of my graduate studies. I would like to thank my mother, Jung-Ji who has been the best listener for every new approach and idea that I discussed with her while paying a huge amount for these international calls. I'd also like to thank my brother, Sangwoo for his support during my hard time.

I would like to thank all of my colleagues in Community Grid Laboratory and friends of mine.

TABLE OF CONTENTS

List of Figures

List of Tables

ABSTRACT

The enormous growth in wireless communications and miniaturized handheld devices in the last few years, have given rise to a vast range of new services for heterogeneous user environments. The concept of a Peer-to-Peer (P2P) Grid has extended traditional distributed services to accommodate diverse user devices, resources sharing environments and higher level services. In traditional multi-tier distributed services, services have generally been designed for accessing back-end resources and middleware without taking into consideration the display and networking capabilities of clients. Current developments in the design of miniature devices, their growing compute, display and communication capabilities, combined with their increasing ubiquity in day-to-day life mandate a paradigm shift in the way services are designed.

In this dissertation, we suggest that the nature and capabilities of these devices be factored in, into the design of services. Doing so would enable the service to cope with the ever-changing landscape of pervasive devices within the P2P Grid infrastructure. We address a number of interrelated issues. First, we propose a software architecture, called the *Modular Data Pipelining Architecture* (MDPA), which separates user presentation from data, and refines data processing within stages of the pipeline which

can potentially be deployed for Web-based collaborative application in P2P Grid environments. Second, MDPA provides a modular approach to this problem which can be expanded incrementally to deal with future changes in the nature of these devices. Third, although this thesis has been organized in the context of device capabilities, some of the ideas could be extended to deal with changing protocol, transport and communication standards in other network centric communication environments.

We also introduce how we deploy the software architecture for the P2P Grid represented with Web service semantics. We also present a Universally Accessible Web service architecture, *CAROUSEL Web service*, which is our collaborative Grid service linked with a Web Service infrastructure and event brokering service. Finally, we also describe our approaches to content adaptation for different devices and users.

CHAPTER 1 INTRODUCTION

The explosive growth of the wireless network in the last few years and emergence of new devices has given rise to a vast range of new services for the heterogeneous user environment. The concept of Peer-to-Peer (P2P) Grid [Fox+02a] allows the distributed system to accommodate the diverse user devices and higher level of services. Hitherto, traditional multi-tier distributed services have generally been designed considering only back-end resources and middleware. The computing, display and networking capabilities of clients were not taken into consideration. Clients were expected to meet a set of explicit, and sometimes implicit, constraints prior to utilizing the service. The current developments in the design of miniature devices, their growing compute, display and communication capabilities, combined with their increasing ubiquity in day-to-day life mandate a paradigm shift in the way services are designed. We suggest that the nature and capabilities of these devices be factored in, into the design of services. Doing so would enable the service to cope with the ever-changing landscape of pervasive devices for P2P Grid architecture.

There have been various approaches to minimize the effects of interface changes on the application as a whole, and promote portability among heterogeneous user display environments. In early 80s and 90s, there have been the classical approaches of model-based User Interface Management Systems (UIMS) according to the Seeheim model [Phaff+85], which emphasized the separation between application logic, dialog control, and presentation component. Model-View-Controller (MVC) pattern [Goldberg+84] also

addressed the separation of data and user view, in addition, enabled multiple views to be rendered individually from the data and communicate each other synchronously. The enormous approaches followed such as the Presentation-Abstraction-Control (PAC) model [Coutaz+87] and Arch model [Bass+92]. Also various software architectures combined multiple conceptual models have been appeared [Dewan+95, Coutaz+97]. The paradigm of separation with data and user presentation is deployed in current software architectures designed for scientific visualizations [Brodlie+96], and also web-based applications [Shaeck+02, Arsanjani+02]. Nevertheless, existing software architectures do not adequately meet the needs of pervasive collaboration services in P2P Grid. In design of collaborative services supporting heterogeneous devices, there are various requirements should be considered. For instance, multi-functional collaborative services such as require more general software architecture to define and extend their features flexibly including asynchronous/synchronous collaboration. Also it expects to adapt heterogeneous user devices and resources. Software architectures designed for specific system such as scientific visualization is not suitable for general purpose services. Meanwhile, classical conceptual patterns are not enough to define current sophisticated distributed services which have multiple steps of data processing for user presentation, although the basic concepts of them offer good basis for structure new software architecture.

In this thesis, we propose software architecture, called the *Modular Data Pipelining Architecture* (MDPA), which inherits the idea of separating user presentation from data, and refines data processing as stages of the pipeline for Web-based collaborative application in P2P Grid environments. MDPA provides more extensibility and interoperability to P2P Grid. Therefore, it enables P2P Grid to support heterogeneous user devices with flexible resource accessing. MDPA supports collaboration between users with various flexibility of resource sharing.

A modular approach of MDPA would facilitate re-use and would only entail small incremental changes to cope with new devices. This thesis presents a modular architecture for designing services in the context of the P2P grid, which as we will elaborate in subsequent sections, is an emerging system that draws upon the body of work existing in P2P and grid systems. This modular approach builds upon the emerging

standards based approach to building, composing, registering and discovering services [Hoscheck+02].

The impact and scope of this thesis are at three distinct levels. First, it suggests changes in the way services are designed. Second, it outlines a modular approach to this problem which can be expanded incrementally to deal with future changes in the nature of these devices. Finally, although this thesis has been organized in the context of device capabilities, some of the ideas of this thesis could be extended to deal with changing protocol, transport and communication standards with its software architectural idea.

1.1 Introduction

The portability of new miniaturized devices, together with their ability to connect conveniently to networks in different places, makes *mobile and ubiquitous computing* possible. Mobile computing is the capability to perform computing tasks while a user is on the move, or visiting places other than the usual environment [Coulouris+01]. Similarly, ubiquitous computing is the harnessing of many small, cheap computational devices that are present in users' physical environments, including the home, office and elsewhere. The term '*ubiquitous*' is intended to suggest that small computing devices will eventually become so pervasive in everyday life that they are scarcely noticed. That is, their computational behavior will be transparently and intimately tied up with their physical function. There are several traditional approaches to mobile computing, such as, mobile agents, network computers, or thin-client concept as variations of the traditional client-server model. These approaches are considered by the system developers, based on their characteristics and constraints of services.

Currently, popular Internet services supports diverse devices. Yahoo! [Yahoo] currently provides services for PDAs and Web-enabled phones with features to access search engines, and send e-mail, in addition to remote synchronizing services. AOL [AOL] and MSN [MSN] also support mobile devices with their well-known instant messenger and access to e-mail accounts. Initially, their services were dominated by the

portal service approach. However, these services have evolved to encompass richer services in real time environment.

Distributed services for mobile devices over the Internet are getting increasingly sophisticated and networked resources are innovatively utilized. The resources in distributed services include database accesses, online services, remote file services, and search and subsequent discovery of a communal resource located on another computer. Emergence of new network communication architectures has influenced resource utilization. The most significant innovative architectures are the emergence of these Peer-to-Peer technologies and Grid systems.

The Grid [Grid, GridWG, Foster+98, Globus] has made dramatic progress recently with impressive technology and several large important applications initiated in high-energy physics [PhysicsGrid, IVDGL], earth science [NEES, SCEC] and other areas [Johnston+00, Fusion]. At the same time, there have been equally impressive advances in broadly deployed Internet technology. We can cite the dramatic growth in the use of XML, the "disruptive" impact of peer-to-peer (P2P) approaches [Oram+01] that have resulted in a slew of powerful applications and the more orderly, but still widespread, adoption of a universal Web Service approach to Web-based applications [Booth+03, Christensen+01]. There are no crisp definitions of Grids and P2P Networks that allow us to unambiguously discuss their differences and similarities and what it means to integrate them. However these two concepts conjure up stereotype images that can be compared. Taking "extreme" cases, Grids are exemplified by the infrastructure used to allow seamless access to supercomputers and their datasets. P2P technology facilitates sophisticated resource sharing environments between "consenting" peers over the "edges" of the Internet, enabling ad hoc communities of low-end clients to advertise and access resources on communal computers. Each of these examples offers services but they differ in their functionality and style of implementation.

The P2P example could involve services to set-up and join peer groups, browse and access files on a peer, or possibly to advertise one's interest in a particular file. The "classic" grid could support job submittal, status services and access to sophisticated data management systems. Grids typically have structured robust security services while P2P networks can exhibit more intuitive trust mechanisms reminiscent of the "real world". Grids typically offer robust services that scale well in pre-existing hierarchically arranged

organizations. P2P networks are often used when a best effort service is needed in a dynamic poorly structured community. If one needs a particular "hot digital recording", it is not necessary to locate all possible sources; a P2P network need only search enough plausible resources to ensure that success is statistically guaranteed. On the other hand, a 3D simulation of the universe might need to be carefully scheduled and submitted in a guaranteed fashion to one of the handful of available supercomputers that can support it. There are several attractive features in the P2P model, which motivate the development of hybrid systems. Deployment of P2P systems is entirely user driven obviating the need for any dedicated management of these systems. Resource discovery and management is an integral part of P2P computing with peers exposing the resources that they are willing to share and the system (sometimes) replicates these resources based on demand. Grids might host different persistent services and they must be able to discover these services and the interfaces they support. Peers can form groups with the fluid group memberships and are thus very relevant for collaboration [Fox+02]. This is an area that has been addressed for the Grid in [Grid] and also in a seminal paper by Foster and collaborators [Foster+98] addressing broad support for communities.

A P2P Grid would comprise services that include those of Grids and P2P networks while naturally supporting environments that have features of both limiting cases. We can discuss two examples where such a model is naturally applied. In High Energy Physics data analysis (e-Science problem discussed in [eScience]), the initial steps are dominated by the systematic analysis of the accelerator data to produce summary events roughly at the level of sets of particles. This Grid-like step is followed by "physics analysis", which can involve many different studies and much debate between involved physicists regarding the appropriate methods to study the data. Here we see some Grid and some P2P features. As a second example, consider how one uses the Internet to access information – in search of either news items or multimedia entertainment. Perhaps the large sites like Yahoo, CNN and future digital movie distribution centers have a Grid like organization. There are well-defined central repositories and high performance delivery mechanisms involving caching to support access. Security is likely to be strict for premium channels. This structured information is augmented by the P2P mechanisms popularized by Napster with communities sharing MP3 and other treasures in a less organized and controlled fashion. These simple examples suggest that whether for

science or commodity communities, information systems should support both Grid and P2P capabilities. The proposed P2P grid, which integrates the evolving ideas of computational grids, distributed objects, web services, P2P networks and message oriented middleware, comprises resources such as relatively static clients, high-end resources and a dynamic collection of multiple P2P subsystems. We investigate the architecture, comprising a distributed brokering system that will support such a hybrid environment. Services can be hosted on such a P2P grid with peer groups managed locally and arranged into a global system supported by core servers. Access to services can then be mediated either by the "broker middleware" or alternatively by P2P interactions between machines "on the edge". The relative performance of each approach (which could reflect computer/network cycles as well as the existence of firewalls) would be used in deciding on the implementation to use. P2P approaches best support local dynamic interactions; the distributed broker approach scales best globally but cannot easily manage the rich structure of transient services, which would characterize complex tasks. Such P2P Grids should seamlessly provide services and resources to users, which are also linked to each other. We can abstract such environments as a distributed system of "clients" that consist either of "users" or "resources" or proxies thereto. These clients must be linked together in a flexible fault tolerant, efficient, high performance fashion. The messaging infrastructure linking clients (both users and resources of course) would provide the backbone for the P2P grid.

Peer-to-Peer Grids allow heterogeneous user devices to share resources over the edge of the infrastructure. Dealing with heterogeneity in devices, and addressing scaling in this should be inherent in the design decisions for the P2P Grid.

1.2 Key concepts of Peer-to-Peer Grids and Enabling Technologies

P2P Grid is based on the key concepts of distributed object technology, resources available over the Internet and services. Distributed object technology is implemented with objects defined in an XML-based IDL (Interface Definition Language) such as

WSDL (Web Services Definition Language) [Christensen+01]. This allows "traditional approaches" like CORBA [Scallan] or Java [SUN] to be used "under-the-hood" with an XML [Bray+00] wrapper providing a uniform interface. As another key concept of P2P Grid, we consider resources accessible via the Internet. W3C defines any "object" labeled by a URI (Universal Resource Identifier) as a resource, whether an external or internal entity [Berners-Lee+98]. This includes not only macroscopic constructs like computer programs or sensors but also their detailed properties. One can consider the URI as the barcode of the Internet. There are also of course URLs (Universal Resource Locations) that indicate location. One can equate these two concepts (URI and URL). Although inadvisable in principle, it is of course common practice.

Finally, the environments of Peer-to-Peer Grids are built on top of a service model. A service is an entity that accepts one or more inputs and gives one or more results. These inputs and results are the messages that characterize the system. Within the context of WSDL [Christensen+01], the inputs and outputs of services are defined as *input and output ports* as components of the Web services.

Within this architecture, everything is a resource. The basic macroscopic entities exposed directly to users and to other services are built as distributed objects that are constructed as services, enabling access to capabilities and properties through a message-based protocol. Services contain multiple properties, which are themselves individual resources. A service corresponds roughly to a computer program or process; the interface of a communication channel to subroutine calls with input parameters and returned data. The critical difference from the past is that one now assumes that each service runs on a different computer scattered around the globe. In principle services can be dynamically migrated between computers. Distributed object technology allows us to properly encapsulate the services and provide a management structure. The use of XML and standard interfaces like WSDL give a universality that facilitates the interoperability of services from different sources.

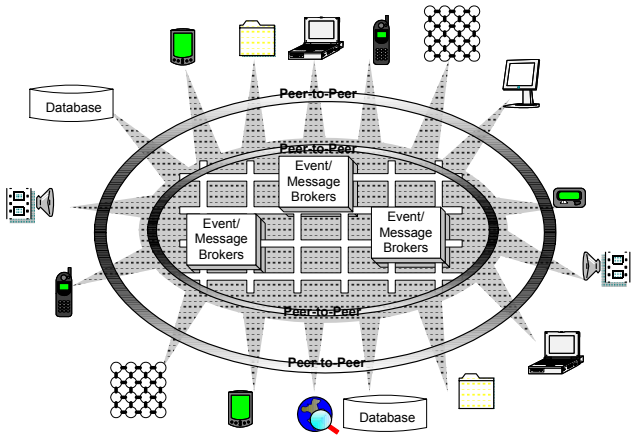


Figure 1.1 A Peer-to-Peer Grid

There are several important technology research and development areas, upon which the above infrastructure builds namely,

- Basic system capabilities are packaged as Web Services [Booth+03]. These include security, access to computers (job submittal, status etc.) and access to various forms of databases (information services), including relational systems, LDAP [Yeong+95], and XML databases/files. Network-wide search techniques about web services or the content of web services could also be included here.
- The messaging sub-system between resources that provide addressing functionality, performance and fault-tolerance [Pallickara+03a].
- Toolkits that enable applications to be packaged as interoperable distributed services and "libraries" or more precisely components. Near-term targets include areas like image processing used in virtual observatory projects or gene searching used in bio-informatics [Globus].
- Application meta-data needed to describe all stages of the scientific endeavor.
- Higher-level and value-added system services such as network monitoring, collaboration and visualization.
- Investigation of Semantic Grid [Hendler+02] or approaches to the representation of and discovery of knowledge from Grid resources.

- Aggregator for integrating multiple distributed services and defining the user's display, which accepts user customization and delivers user interfaces [Jetspeed, Websphere].

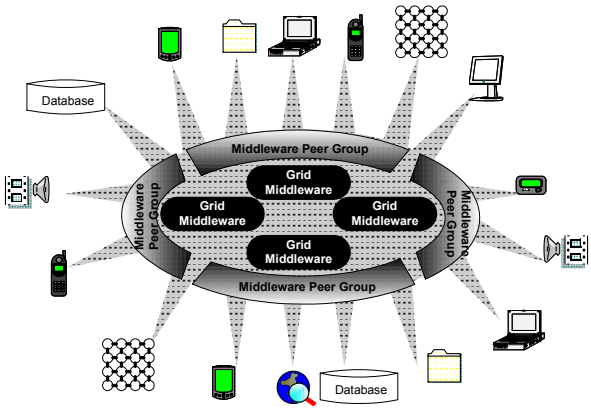


Figure 1.2 Middleware Peer Groups of Services at the "edge" of the Grid

1.3 Building a Distributed Service within a Peer-to-Peer Grids Concept

In a Peer-to-Peer Grids architecture, traditional Grid services with a Web middleware that mediates between back-end resources and clients are arranged as depicted in Figure 1.1. Just as in a classic 3-tier architecture, there are clients, back-end resources and multiple layers of middleware linked with the messaging service. This is a natural virtual machine seen by a given user accessing a resource. However, the implementation could be very different. Access to services could be mediated by "servers in the core" or alternatively by direct peer-to-peer interactions between machines "on the edge". The distributed object abstractions with separate service and message layers allow either Peer-to-Peer or server-based implementations. The relative performance of each approach would be used in deciding on the implementation to use. P2P is best in supporting local dynamic interactions; the server approach scales best globally but cannot easily manage the rich structure of transient services, which would characterize complex

tasks. Figure 1.2 shows Grids control central services while “services at the edge” are grouped into less organized “middleware peer groups”. Often one associates P2P technologies with clients but in a unified model, they provide services, which are part of the middleware. As an example, one can use the JXTA search [SUN+03] to federate middle tier database systems; this dynamic federation can use either P2P or more robust Grid security mechanisms.

We can ask if this new approach to distributed system infrastructure affects key hardware and software infrastructure, and their performance requirements. First we present some general remarks. Servers tend to be highly reliable these days [Fox+02 a]. Typically, they run in controlled environments, but also their software can be proactively configured to ensure reliable operation. One expects servers to run uninterrupted for months and often one ensures that they are modern hardware configured for the job at hand. Clients on the other hand can behave quite erratically, with unexpected crashes and network disconnections, as well as sporadic connectively typical of portable devices. Transient data can be stored on clients; however permanent information repositories must beside on servers – here we talk about “logical” servers as we may implement a session entirely within a local peer group of “clients”. Robustness of servers needs to be addressed in a dynamic fashion and on a scale greater than in previous systems. Traditional techniques – replication and careful transaction processing – can probably be extended to handle servers. Realistically, clients are assumed to be both unreliable and sort of outside our control. Some clients will be “antiques” and underpowered and are likely to have many software hardware and network instabilities. In the simplest model, clients “just” act as a vehicle to render information for the user with all the action performed on “reliable” servers.

1.4 Motivations behind MDPA

Our interest in MDPA is motivated above all by the prospect of innovative new uses of diverse devices and wireless communication services. During the last few years,

mobile computing with pervasive devices has emerged as a new computing paradigm. The key enabling technologies are,

- *Portable devices*: Handheld devices including Personal Digital Assistants (PDAs), laptops, smart phones, pagers, and wearable devices such as JAVA ring, smart watch etc. These devices have diverse computing capabilities such as 66 MHz to 400 MHz, with displays ranging 1bit black and white to over 64K color display. The interactive capabilities include touch screen style interaction with a stylus pointer, small buttons, extensible keyboard, voice recognizing system, etc.

- *Wireless and satellite network communication*: Reliable wireless network communication such as wireless LAN (802.11b) or 3G network communications, has integrated portable devices integrated to the clients of distributed systems over the Internet. Meanwhile, the use of satellite communication has maximized the advantage of mobile computing; also it is joined with wireless computing and provides high quality of networking services. The current broadband wireless networking service is capable of delivering new services – streaming video, audio, gaming, backup storage, and much more. These network service providers also support various middleware services enabling their clients to access existing information systems.

The favorable effects of increasing capabilities on client devices coupled with falling prices have resulted in widespread use of the devices to access information systems. The distributed services of P2P Grid should consider supporting virtually any device, over a broad range of networks, and allow new applications to be easily built and deployed.

As demonstrated in figure 1, the concept of P2P Grid supports natural environments that have back-end resources and diverse users that require the service. Before the emergence of various user devices with wireless network communication, the 3-tier distributed service could expect unified user devices for their clients. Therefore, the infrastructure of the service focused on the back-end resources or middleware messaging services. New approaches that address the heterogeneity of back-end resources have been focused on during the last several years. Object brokering systems, such as CORBA

[Scallan], process different types of objects from various computing resources. However the current diversity of user devices requires the data processing that considers the heterogeneity of end user devices as well.

Early efforts include software architectures for user interface, such as the Seeheim model, MVC, and PAC. After these classical models, there have been various advanced approaches in field of user interface design. One of the most active approaches is scientific visualization, which maps raw data set to a graphical form for effective processing by our visual senses. The scientists are well served by a range of powerful systems, such as IRIS Explorer [Brodlie+96], AVS [Upsen+89] and IBM Data Explorer [IBM+02]. These support a visual programming paradigm of Modular Visualization Environments (MVE), which allows the users to select a set of modules to fit together in a pipeline, transforming raw data to geometry and then to images. Also collaborative capability between scientists is considered in their visualization model [Wood+95]. However, since these approaches are focused on a specific task, presenting scientific data, it is not adaptable into general purposed distributed services without modification. Meanwhile, the concept of separation user view from resources is deployed in emerging Web-based service infrastructures [Arsanjani+02, Shaeck+02]. To support various types of collaborative features, P2P Grids require software architecture refined with suitable granularity and a clear definition of process. With the current portal based infrastructure, we cannot describe sophisticated collaborative features, such as message based synchronous collaboration.

MDPA provides a modular approach to asynchronous/synchronous collaboration in P2P Grids infrastructure. It supports reusability of each data module, thus only entail a small incremental adapting effort for new devices or environments. MDPA is based on the approaches of standardization and emphasizing interoperability. Finally, MDPA supports the universal accessible environment for P2P Grid service to heterogeneous user devices.

1.5 Research Objectives

There are various approaches to universal accessibility to the rich functionalities available in distributed systems. P2P Grids should consider the heterogeneity of environment very carefully. Diversity of network communication is one of the aspects that needs to be considered. The users require seamless access to services over the heterogeneous network environment. Service providers should thus consider the diversity in computing and display capabilities of entities utilizing their services. The capability of computing or displaying is extremely optimized for the characteristics of the devices. In this dissertation, we aim for a universal accessible Peer-to-Peer Grid based on a software architecture.

The goal of this dissertation is to design a generic software architecture that provides:

- *Modulated data processing:* The distributed services over the P2P Grids infrastructure provides various capabilities such as: accessing database systems, using a super--computing unit for computational purposes, maintaining systems remotely, collaborating with other users, and visualizing data from services etc. The data processing in these different tasks different steps of processing leading up to the task to provide service. MDPA is constructed to define generic data processing that is able to consider heterogeneous user environments and various service tasks including collaboration.

- *Flexible Interoperability between data processes:* The interprocessing between objects in a P2P Grid is a fundamental requirement to facilitate multi user services. MDPA is expected to support various resource sharing methods for flexible collaboration. Without the characterization of the data process, the processing cannot predict the types of interaction with other processors in a given step. The characterization of data processing is for avoiding wasted resources derived from ambiguous interface design. In addition, MDPA should provide the means to communicate to other data processing seamlessly.

- *User Interactivity:* If the interprocess communication is for the communication between data processes, we consider communication between users and services. The user interactive system needs the ability to perform the user's request. The request from the user is understood as a user event in an event-based system. To process event-based distributed services, the new architecture should be able to support communications from both sides: from users to back-end services for delivering user's requirements, and from the back-end services to the users for showing them the results.
- *Universal Accessibility:* Adapting to heterogeneous user devices is one of the objectives major in MDPA. Adapting to new environments should be factored into the design of P2P Grid services. Thus, adapting process should be considered even in the design of collaborative features, and it is one of critical factors needed to decide the type of collaboration.

Eventually, the new architecture provides the environment for data processing for heterogeneous clients with diverse devices and supports distributed services including real-time collaboration. This architecture enables the Peer-to-Peer Grid to extend the concept of resources to every heterogeneous computing unit.

We also provide a prototype of this software architecture, which shows the fundamental units and deployment to collaboration models. This prototype also presents how universal accessibility is adapted to this software architecture. Moreover, the prototype will be an example of the implementation over the modern system infrastructure.

1.6 Author's Contributions

This dissertation is based on accumulated experiences in the research of collaborative distributed services. As a graduate student, Sangmi Lee has been working on the project CAROUSEL in Community Grid Laboratory led by Dr. Geoffrey C. Fox, since 2001. She has been active in the area of collaborative distributed service since she started to work for the Tango Interactive project at Syracuse University with Dr. Geoffrey

C. Fox in 1999. Since 2000, she has focused on the universal accessibility issue while working on the Garnet message service system developed at Florida State University. Here is the summary of her contributions to this dissertation,

- Designing and modeling of the event model of the MDPA: She defined each event type and did modeling with Petri Net theory.
- Designing and modeling service extensions to MDPA: She designed and modeled the aggregation and collaboration of services using MDPA.
- Designing the architecture of the CAROUSEL Web service [Fox+02d]: As a major designer, she designed the Carousel Web service architecture.
- Implementation a prototype of the CAROUSEL Web service with the collaborative SVG browser [Lee+03a]: As a prototype, she implemented a collaborative Web service, the collaborative SVG browser for heterogeneous user devices.
- Designing and implementing the application sharing features for mobile devices within the Garnet system [Fox+02b, Fox+02c, Lee+02, Lee+03b]. The Garnet system provides application sharing methods, shared Display and shared export. She designed and implemented both these sharing features for mobile users.
- Designing and implementing the user application for GMSME (Garnet Message Service Micro Edition) [Fox+02b]: She designed and implemented features of shared application for the client with GMSME.

1.7 Dissertation roadmap

This dissertation is organized in the following manner:

In chapter 2, we present the background and discuss previous work in data processing of distributed systems. First, we present existing software architectures for processing user presentations. Then, we focus on the accommodation of heterogeneous devices and multi-user interactive systems.

Chapter 3 describes the fundamental architecture of MDPA and the approaches to satisfy the major design issues. Also, we present how we deploy MDPA into the emerging Web Services infrastructure. Chapter 4 presents our modeling of MDPA. We

utilize the Filter and Pipes pattern to structure, and Modular PT-net theory to represent, MDPA.

In chapter 5, we discuss the aggregation of services and collaborative services with MDPA. Chapter 6 focuses on incorporating content adaptation strategies in MDPA. We present content adaptation for heterogeneous devices with the demonstrative case study of the Carousel Web Service. Based on the modeling in chapter 5, we present how the services provide collaborative features to users in heterogeneous environments in chapter 7. Also, we describe how we deploy the collaboration models within the Web Service infrastructure. Finally, we present a demonstrative case study with the Web services architecture, the CAROUSEL Web Service.

Chapter 8 describes the conclusion of the dissertation and analyzes our approach. Also, we summarize our work and suggest plans for future researchers investigating this area of distributed computing.

CHAPTER 2 BACKGROUND

There have been several software architectures for interactive service. These are also known as user interface architectures or user interface management systems. e.g. Seeheim [Phaff+85], PAC-Amodeus [Coutaz+95], Model-View-Controller(MVC) [Goldberg+84]. Most of these architectures are based on the traditional approach of an interactive software, which separates the user interface from the application. The application part processes the task related to the functionality of the service. Meanwhile, the user interface part contains the representation of this functionality to the user(s) of the system.

The principle of separating an interactive service into application and user interface parts has advantages while trying to achieve universal accessibility. Such a scheme provides demarcation of the device dependent part and the functional part of a service. However, the types of devices that can participate in services, which were offered only for traditional desk top systems before, have increased with enhanced capabilities in miniaturized machines and advances in wireless communications. Therefore, the services cannot support the diversity of participants' devices with the traditional user interface. Finally, there are several efforts to provide the customized services to users who access the service with CPU/bandwidth limited devices, such as supporting adapting content or mobile protocols in a proxy architecture.

In this chapter, we review well-known traditional software architectures in section 2.1. In section 2.2, we describe some adaptive architecture which extends these

traditional architectures. In section 2.3, we describe popular adapting technologies supporting the software architecture for heterogeneous environments. Finally, we introduce the Web service infrastructure briefly in section 2.4. Web Service infrastructure provides interoperability between Web-based applications with its standardized data and interface design. We use the Web service infrastructure for prototyping the MDPA architecture in this dissertation.

2.1 Classical Software Architectures for Presentation Generation in User Interactive Services

Among the many existing user interface architectures, we discuss the PAC-Amodeus [Coutaz+95] and the MVC [Goldberg+84] architectures. One of the merits of these architectures is that these architectures can be mapped very well onto object-oriented concepts.

2.1.1 PAC-Amodeus architecture

The PAC-Amodeus model [Coutaz+95] is a combination of the Arch model [Bass+98] and the Presentation-Abstraction-Control (PAC) model [Coutaz+87]. The Arch model, which refined the Seeheim model [Phaff+85], distinguishes five components of an interactive service. The structure of the Arch model is shown in Figure 2.1.

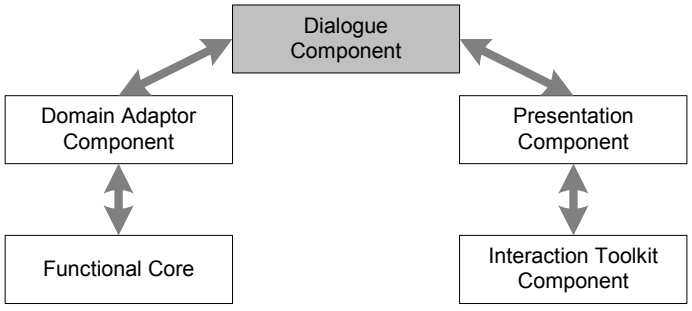


Figure 2.1 Arch model

The *domain-specific component* encapsulates the functionality of the system and offers an abstraction of the application semantics. The *interaction toolkit component* encapsulates the physical interaction, such as the platform-dependent implementation of user interface widgets as well as hardware details. The *presentation component* provides an abstraction from the interaction toolkit component and provides platform-independence. The *dialogue component* contains task level sequencing and provides the mapping between domain specific objects and user interface specific objects. The *domain adapter component* adds domain related behavior to the functional core that is needed by the dialogue component.

There are three kinds of objects depending on the path that it is traversed. The *domain object* traverses through the dialogue component, domain adapter component, and functional core. The dialogue component and the presentation component exchange *presentation objects* (abstract, virtual objects representing aspects of interaction). The presentation component and the interaction toolkit component exchange *interaction objects*.

The Presentation-Abstraction-Control (PAC) model is used [Coutaz+87] for the dialogue component. PAC is a multi-agent model that structures the dialogue component as a hierarchy of interacting agents. For example, an agent can correspond to a window, a group of widgets, or a single widget.

There are three facets in each agent: an *abstraction* facet, a presentation facet, and a control facet. The abstraction facet contains the data or objects and the *presentation* facet encapsulates the presentation logic of the agent. The *control* facet controls the communication between abstraction and presentation and the communication between subordinate and superordinate agents.

The PAC-Amodeous model separates the functional part and the user display part from an application. However, since the functional parts are encapsulated as a part, the design of integrating new adapting functionality is not clearly defined in the whole architecture. This can lead to raising complexity for maintaining a service, such as adding content adapting technology for new emerging device.

2.1.2 Model-View-Controller architecture

The Model-View-Controller (MVC) architecture is a well-known object-oriented model that has been applied earlier in the Smalltalk environment [Goldberg+84]. As one of three major components, the *model* represents the underlying information of a specific user interface element. The *view* displays this information in a certain way, while the *controller* knows how the user interactions with the view will affect the information in the model. If the model changes, it notifies its view(s) of this change. The model itself is linked to the application objects. The view and the controller are tightly coupled and they are even often combined into a single object. Note that there is a lot of coupling and interdependency in instantiations of the MVC architecture.

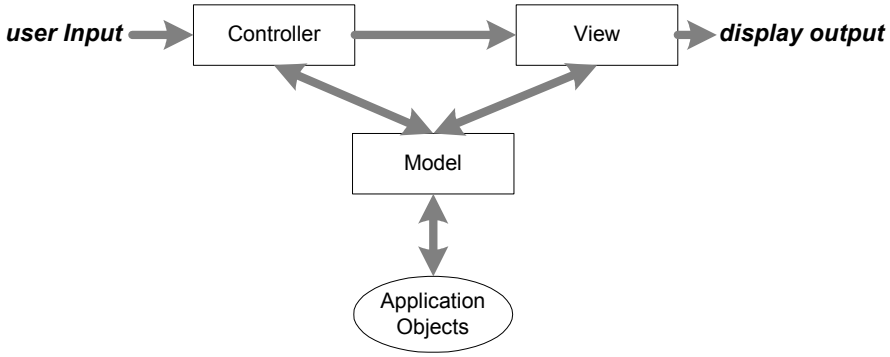


Figure 2.2 The MVC architecture

Figure 2.2 depicts the MVC architecture. The MVC architecture makes multiple synchronized views on the same information possible, which is required in heterogeneous environments. Furthermore, application objects are isolated from changes in the 'look and feel' of the user interface. Therefore, it can be an object which is shared in a collaborative service.

However, the MVC architecture also has the adaptability problem. The concept of adaptability is usually defined as "robustness to change". Since the MVC architecture is refined coarsely, the design of an adapting process for new devices cannot be clearly defined. Therefore, adapting a new device or new technology will require a lot of effort in the maintenance of the service.

2.2 Approaches to Adapt New Devices

Adaptation for data processing for heterogeneous clients has been actively investigated by many research groups. In this subsection, we explore the approaches based on the categories of, proxy based architectures, standardization of data and protocols, and transcoding technologies.

2.2.1 Proxy based Architectural Approaches

One of the most common approaches is developing a *proxy-based* system for new client environments. Here the client environment includes new hardware, new network communication technology, a new security method and so forth. A proxy is defined as a service representative that resides at the client's site [Smith+99]. It provides an interface to the service and takes care of the communication protocol with that service. It also takes care of the marshalling of data, checking of the validity of calls, and any other low-level processing, thereby making distribution transparent to the client. Due to this transparency proxies are popular and have been adapted by many distributed systems.

The proxy service provides seamless access from heterogeneous network communication environments. iMobile from AT&T Wireless [Rao+01] is a proxy-based platform, which acts as a message gateway that allows mobile devices using various protocols on different access networks to relay messages to each other.

The WBI project [Barrett+99] from IBM uses intermediaries to produce and manipulate web content, perform content distillation, and implement protocol extensions. Recently, a transcoding proxy was introduced as a web intermediary between Web servers and client devices to adapt to the varying bandwidths of different client communication links. Similarly, TranSend [Fox+98], a scalable transformational Web Proxy from UC-Berkeley, focuses on how datatype-specific content distillation can be efficiently implemented by deploying infrastructural proxy services. iMobile from AT&T [Rao+01] and the ICEBERG project from UC-Berkeley [Wang+00] concentrate on any-to-any communication services and personal mobility services.

Moreover, W3C working group called CC/PP (Composite Capabilities/Preferences Profiles) [Reynolds+00] develops a structured and universal format that allows a client device to tell an origin server or proxy about its profile.

The proxy-based architecture has the advantage of being more manageable while providing an easy scheme to extend existing services for heterogeneous devices. However, a proxy may not always be the best place to perform content adaptation. The scheme may entail in efficient bandwidth utilization if a lot of contents downloaded to the proxy are later discarded due to the limited screen size or bandwidth of the receiving device. Furthermore, the content provider may prefer to perform its own transcoding service to fully control the outcome. The Apache Cocoon [Apache+99] project allows the automatic generation of HTML, PDF, and WML files through the processing of statically or dynamically generated XML files using XSL [Adler+01] and XSLT [Kay+03].

2.2.2 Standards for Heterogeneous Environments

With a profound impact of XML [Bray+00], the information over the Internet is powered with the interoperability. An important feature of this language is the separation of presentation from the content, which makes it easier to select and/or reformat the data. XML is used as an extensible syntax for metadata model designed for interoperable information systems. The markup languages derived from XML also inherit this interoperability of XML.

Apart from usability, interoperability is one of the critical topics for integrating heterogeneous application running of different platforms. Standards are governed by many different groups including official bodies, not for profit organizations and industry consortia. Each group is charged with gaining consensus for the implementation of particular standards, often within a particular application domain. For example The World Wide Web Consortium (W3C) is the consortia leading development of standards relating to the Web. W3C provides guidelines for developing universal accessible Web content [Chisholm+99], and detail technical resources such as [Koivunen+99, Jacobs+99, McCathieNevil+00].

OASIS (formerly SGML Open, the international consortium that has guided the SGML industry since 1993) provides the guidance, process, and infrastructure necessary for e-business framework, Web Services, security, public sector etc [OASIS+03]. There are also several groups developing standards for specific areas: Multimedia Services Affiliate Forum (MSAF) [MSAF], International Digital Enterprise Alliance (IDEAlliance) [IDEAlliance], the European Telecommunications Standards Institute (ETSI) [ETSI], Text encoding initiative (TEI) [TEI], Unicode Consortium [UnicodeConsotium], and American National Standards Institute (ANSI) [ANSI], etc.

For implementers it is as well to remember that there are an awful lot of standards being developed by a very wide range of consortia and official bodies. This has meant that often standards are being developed, by many different groups, for products and services that essentially offer the same functionality.

XML is deployed for developing metadata sets. Berners-Lee defines metadata as "machine understandable information for the web" [Berners-Lee+97]. W3C has developed the Resource Description Framework (RDF) [Lassila+99] and its relative Platform for Internet Content Selection (PICS) [Chu+98]. Dublin Core Metadata Initiative (DCMI) [DCMI] develops interoperable online metadata standards for general purpose of resources. DCMI's activities include metadata refinements and guidelines for the metadata usage. The idea of metadata to manage the resources on the Web is developed to ontology-based knowledge representation languages. The term ontology is a formal specification of the concepts and relations of some domain. Structuring resources over the Internet is approached with underlying information structure based on metadata represented with XML. The higher lever information structures make it possible for the heterogeneous devices to manage the information content from the distributed system in unified way.

Obviously, using XML enabled the heterogeneous applications to exchange information each other and build distributed service effectively. It also encompasses applications developed for different platforms. However, as we mentioned before, the standardizing is investigated in a lot of groups in the world. It causes another interoperability problem also. According to the system designer's choice, the interoperability can be limited within the group of applications using same metadata sets.

Moreover, the complexity of data processing such as parsing, indexing, or searching documents can overload some of the limited CPU capability handheld devices.

2.2.3 Incorporating with Transcoding Technologies

Transcoding is the process of dynamically mutating and customizing a document based on the characteristics of the requesting device and on the user's preferences. Transcoding is used to convert image or video formats (reduction resolution or data compression). However it is also used to fit document and graphics files to the unique constraints of mobile devices and other Web-enabled products.

Web clipping technology proposed by Palm [Palm+02] initially is now the widespread solution for accessing to Web contents from miniaturized wireless devices. The web clipping architecture includes client-side applications that run on a handheld device, proxy servers for handling translation between the web clipping application format and HTML, and content servers. The client-side application is called a web clipping application. It is constructed in HTML and translated into the web clipping application format. Once this application is installed on a handheld device, content is delivered from the content provider's own web site as a subset of the HTML 3.2 standard.

A number of distributed services use the transcoding technology to generate the document for their heterogeneous clients. Duke University's Quality Aware Transcoding project [Chandra+00] and TranSqui [Maheshwari+01] from University of Massachusetts utilize the transcoding technology in their proxy server and provide accessibility to heterogeneous user devices.

As a case study, we describe how IBM's *WebSphere Everyplace Access* [IBM+02] applies the transcoding technologies to their Web service based architecture. We will discuss the Web service infrastructure in detail, in section 2.3.

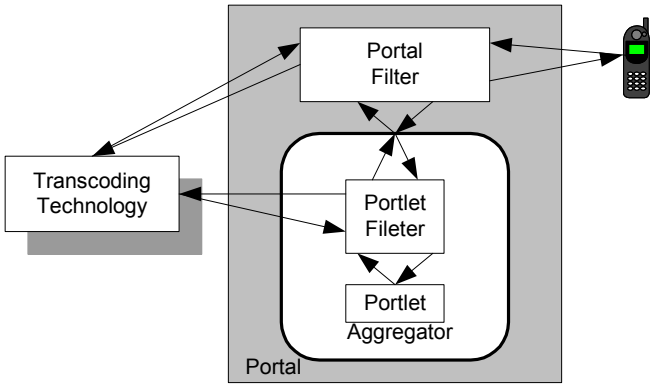


Figure 2.3 WebSphere portal and transcoding technology

WebSphere Portal invokes transcoding technology at two different levels: the individual portlet level and the full page level as depicted in Figure 2.3. Meanwhile, WebSphere defines the portlets as the visible active components end users see within their portal pages [WebSphere].

The client device makes a request to the portal, which the Portal Filter receives. The Portal Filter calls the WebSphere Portal to receive its contents, which then calls the aggregator to aggregate the page. Then, the aggregator selects the portlets based on a combination of values. If a portlet is configured to be transcoded, and it provides a markup that the Transcoding Technology can use, then it will be selected. The portlet filter calls the portlet to receive its contents. The invoked portlet then returns its contents. The portlet filter sends the portlet's contents to Transcoding Technology for processing, associating the portlet's contents with a default URI unless otherwise specified. The Transcoding unit then returns the processed contents to the portlet filter. The portlet filter returns the processed contents as if it were a proper portlet returning contents normally. The aggregator, having finished aggregating the page, returns the page.

Now, the portal filter sends the portal's aggregated contents to Transcoding unit for processing. Transcoding unit returns the processed contents. The portal filter returns the processed contents to the client. Transcoding technology is adapted to the Web service

architecture as a computing unit and finally the user gets the regenerated document fitted on their environment.

2.3 Challenges posed by Web service Architecture

W3C's Web service working group defines Web service in [Booth+03],

> *"A software application identified by a URI, whose interfaces and bindings are capable of being defined, described, and discovered as XML artifacts. A Web service supports direct interactions with other software agents using XML based messages exchanged via internet-based protocols"*

The use of Web Services on the World Wide Web is spreading rapidly, because the need for application-to-application communication and interoperability grows. These Web Services provide a standard means of communication among different software applications, running on a variety of platforms and/or frameworks. Therefore, the Web Services enables us to deploy our software architecture efficiently with its well-defined interfaces and communication strategies in this dissertation.

Web Services is enabled based on the potential for a combination of XML [Bray+00], the Web, the SOAP [Box+00] and WSDL specifications [Christensen+01], and to-be-defined protocol stacks to address many of the problems these technologies have encountered. Compared to previous distributed object systems such as Microsoft's COM family and the OMG CORBA standard [Scallan], Web Services provides interoperability between processes over the Web.

The Web services infrastructure places into relationship various components and technologies that comprise a Web services "stack" or completely functional implementation. Valid implementations include subsets or parts of the stack, but must at least provide the components within the basic architecture. Components and technologies that extend the basic architecture are represented within the extended architecture.

The basic architecture includes Web services technologies capable of:

- Exchanging messages
- Describing Web services

- Publishing and discovering Web service descriptions

The basic Web services architecture defines an interaction between software agents as an exchange of messages between *service requesters* and *service providers* as illustrated in Figure 2.4. Requesters are software agents that request the execution of a service. Providers are software agents that provide a service. Agents can be both service requesters and providers. Providers are responsible for publishing a description of the service(s) they provide. Requesters must be able to find the description(s) of the services.

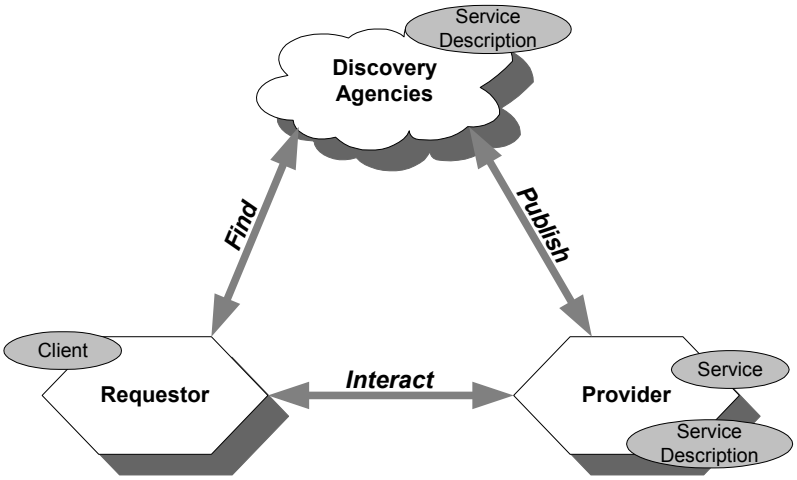


Figure 2.4 Service Oriented Architecture of Web Services

Software agents in the basic architecture can take on one or all of the following roles:

- Service requester -- requests the execution of a Web service
- Service provider -- processes a Web service request
- Discovery agency -- agency through which a Web service description is published and made discoverable

A software agent in the Web services architecture can act in one or multiple roles, acting as requester or provider only, both requester and provider, or as requester, provider, and discovery agency. Since the service description is required to establish a binding, a service is invoked after the description is found.

The basic Web service architecture is based on the interactions between these three roles of agents. The interactions involve the *publish, find*, and *bind operations*. These roles and operations act upon the web service artifacts: the web service software module and its description. In a typical scenario a service provider hosts a network accessible software module (an implementation of a web service). The service provider defines a

service description for the web service and publishes it to a requestor or service discovery agency. The service requestor uses a find operation to retrieve the service description locally or from the discovery agency (i.e. a registry or respository) and uses the service description to bind with the service provider and invoke or interact with the web service implementation. Service provider and service requestor roles are logical constructs and a service may exhibit characteristics of both.

Requesters and providers interact using one or more message exchange patterns (MEPs) that define the sequence of one or more messages exchanged between them. A service description is hosted by a discovery service to which a provider publishes the description, and from which the requester discovers the description. The description includes data type and structure information, identifies the MEP, and contains the address of the service provider.

2.4 Summary

The traditional approaches of interactive software have been focused on separation of the user interface from the application. As the requirement of adapting various devices for the distributed services grows, the software architecture needs to consider the heterogeneity of user devices. In this chapter, we explored the traditional software architecture for the interactive services, and the approaches for the adapting new devices. Additionally, we introduced the Web Service infrastructure which is the emerging interoperable environment for applications based on the Web.

CHAPTER 3 MODULAR DATA PIPELINING ARCHITECTURE

An object in a distributed service is typically in some process structured with several services as depicted in Figure 3.1. We describe the set of distributed services using data pipelines. Each stage of the pipeline is a distributed service with data flowing from one stage into another.

The modularization process is essential to simplify and enhance the adaptability of sophisticated distributed services. Compared to traditional environments, adapting to the needs of pervasive devices requires additional data processing and information. Furthermore, technologies that adapt content have matured significantly and users expect more customized services on their preferred devices.

Our aim in this chapter is to introduce our architectural approach for adapting new devices, and clients efficiently: the Modular Data Pipelining Architecture (MDPA). MDPA abstracts data processing into several stages. Each stage is designed to function independently as a module of the pipeline.

Figure 3.1 Object Processing in Distributed System

Prior to describing the architecture, we outline the basic constraints for this architecture,

- *Modularity*: A service is required to be modularized based on the characteristics of each stage. To satisfy this requirement, each module must perform its task without communicating with other stages, except for processing via predefined input and output ports.

- *Adaptability*: An existing service should be able to adapt new types of resources, including computing units, networking constraints, user devices, and data processing technologies into its workflow with minimum modification. The adapting should be considered from first phase of design architecture as a general data processing not as an additional computing unit.

- *User interactivity*: The system should process user events correctly in the data processing environment designed for heterogeneous user devices.

- *Flexibility*: To support various types of user event processing, the software architecture must enable the service access to any stage flexibly and process user events. This feature is also needed for interoperating between different pipelines.

- *Reusability*: Even for a given user, we should be able to manage data in different ways. For example, if a user's preferences change, the system should reproduce the object based on the new preference. Each stage of MDPA should keep its object instance, and avoid wasting resources on duplicating processing. Furthermore, different services should be able to utilize a stage defined in other pipeline originally.

- *Scalability*: Each application provides an individual data pipeline to generate user presentation. This pipeline must be able to integrate other pipelines and generate a new user display correlated with other pipelines.

- *Interoperability*: For the collaborative system, data pipelines are required to be able to share their objects. Furthermore, integrated distributed services need interoperability between local or remote applications. Our work also addresses this integration of data pipelines with efficient interoperability.

This chapter is structured as follows. Section 3.1 presents overview of MDPA with the characteristics of each stage. In section 3.2, we explain the event processing of MDPA. Section 3.3 describes the basic architecture of a stage in MDPA. In section 3.4 we discuss the design issues in MDPA. We describe the Web service approach to design the MDPA in section 3.5. Finally, in 3.6, we present summary of the chapter.

3.1 Architecture Overview

The fundamental unit of MDPA is the stage, which are the bubbles in Figure 3.2. Every stage is defined based on the characteristics of data processing in that stage. Well-defined stages are helpful for interacting with other pipelines. This interaction with other pipelines requires interfaces to access the object in collaborative service. Without defining the characteristics for stages, interactions between multiple stages and services can be ambiguous and redundant. Even for data pipelines used in different services, developers should be able to reuse a stage which was designed for other services. To facilitate such re-use, the design of a stage should clearly expose its characteristic functionality. Thus, we distinguish five fundamental stages based on their functional characteristics: Data source (DAT), Control Logic (CTL), Document Transformer (DTX), Presentation Client (PCL), and Minimal Client (MCL).

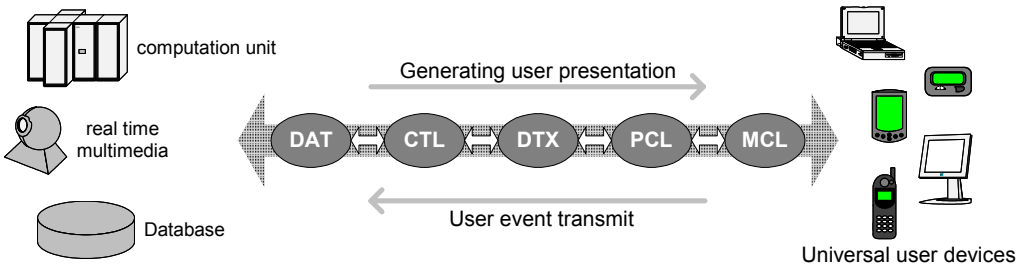


Figure 3.2 Object pipelining in MDPA

3.1.1 Data source (DAT)

MDPA processes various types of data objects from resources sitting on the edge of the P2P Grid: raw data from runtime equipment, output of a computational unit, or output from DBMS etc. The most critical functionality of the DAT stage is accessing these actual resources and passing these objects to the CTL stage.

3.1.2 Control Logic (CTL)

The major goal of CTL is generating an environment for semantic control of the data object in the DAT stage. The raw data in the DAT stage is transformed to a more accessible object type, such as a DOM tree. The user event captured from an end user is transmitted through several stages with filtering, and in the CTL stage, accesses, the original data object with the semantic meaning.

3.1.3 Document Transformer (DTX)

Document Transform is the process of filtering the object, with loss of fidelity, to appropriately fit the limitations of a client device or reduced network bandwidth. The object can be transformed across different data types or similar data types. For example, the image data object can adjust its color depth or the resolution of the image for different devices or needs. Moreover, the image object can be transformed to a completely different data type such as voice or text for extremely limited devices or special needs.

The DTX stage also incorporates data optimizations for limited network environments or data transformations for different network protocols. Therefore, DTX should adapt to the various technologies available locally or remotely.

3.1.4 Presentation Client (PCL)

PCL is the stage of generating an abstract presentation for each client. PCL can be described with the markup languages defining User Interface such as HTML, WML, or UIML. PCL provides a functional interface for clients, and filters the user's event to the user interface event. One of the most important goals of PCL is integrating services. For the integrated service providing multiple features, presentation views from different data pipelines for different services are integrated in PCL stages and provide user presentations in a single user display.

3.1.5 Minimal Client (MCL)

MCL is the stage most dependent on devices and user preferences. Heterogeneity of the input mechanism is considered directly in the MCL. Therefore, each type of device should have its own MCL stage which supports its capability. To facilitate adapting new devices, the MCL stage, which must be located on user's device, provides only minimal functionality -- such as user interface. MCL stage directly deals with user's input and output interfaces, such as framebuffer, key inputs, voice input, etc. In MCL, user events do not have the semantic meaning of the service.

We described the characteristics of each stage in MDPA briefly. In section 3.2, we present the event processing in MDPA.

3.2 Event Processing

MDPA is an event based software architecture. The event-based user interactive system processes data when a user event occurs. The user event is processed with several steps. First, a user event is captured. Here, the user event includes a mouse click, a voice command, keyboard input, etc. The captured event traverses the MDPA data pipeline. Before being delivered, the event is adjusted by the filters defined in each stage. Finally the event is processed and a newly generated user presentation is delivered to the user after traversing the stages along the path from user to certain stage. The user presentation is also tuned in each stage with its presentation filters. Eventually, the user receives a response from the service.

3.2.1 Data flow of Event Processing

There are two directions of dataflow in MDPA,

- Event transmission: This pertains to the path taken by a user event captured in the MCL stage. This user event traverses the data pipeline until the event is processed. Each stage evaluates the user event, modifies it as necessary, and passed it to the next stage.
- Presentation Generation: This pertains to the path taken by the user presentation generated by the event processing. The user presentation is delivered to a user device. Stages that the presentation visits filter the presentation and pass the output on to next stage.

Event transmitting and presentation generating traverse in opposite directions along the data pipeline direction in their dataflow. The MCL stage is the first stage along the event transmission path, while it is the last stage for the presentation generation.

The process of adapting content is bi-directional along the pipeline and performed during both the dataflows. During the session of a service, the user event is captured in the user display environment customized for a specific user device. Since the presentation is filtered, information regarding the presentation cannot be applied to the original data object directly. Therefore, the event transmission also requires adjusting information pertaining to the user event. We call this “back channel communication”.

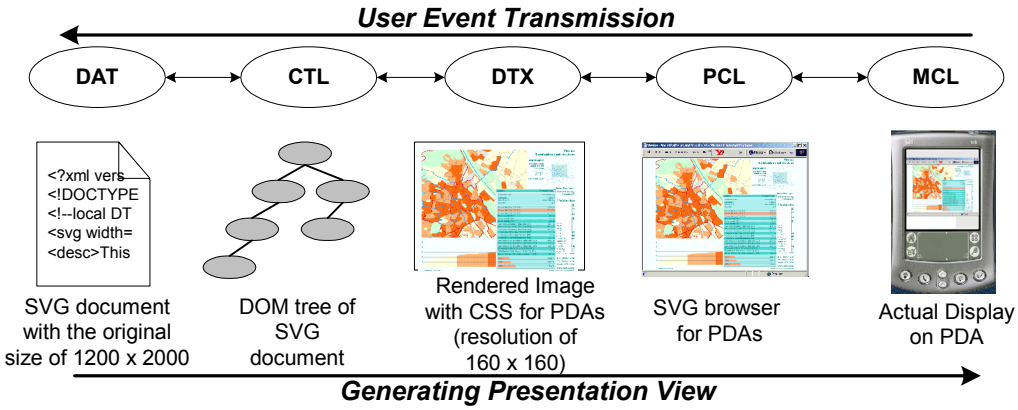


Figure 3.3 Event processing in MDPA with an example of SVG

As an example, we illustrate the Scalable Vector Graphics (SVG) [Ferraiolo+01] browser with a MDPA architecture. SVG is the W3C 2D graphic standard language in XML. SVG supports vector graphic shapes, image, and text. Graphical objects can be grouped, styled, transformed and composed into previously rendered objects. To browse a SVG image, users require a SVG browser, which parses the document, styles and renders the image. Figure 3.3 depicts the event processing in our SVG example. Let's consider an example: drawing an arrow on a map illustrated with SVG documents. The display that the PDA user is seeing currently is a customized view for a PDA at a resolution of 160 x 160. The user points to (80, 80), the center of the presentation in his PDA screen, with the stylus. The event with the information of (80, 80) should be mapped to the position that denotes the center of the SVG image, based on the original coordinate system. Otherwise, the pointed position is not correct in different customized presentation views as depicted in Figure 3.4. Although the user wants to point to the center of the SVG document, the position pointed by PDA client is only a corner of that SVG image in the user presentation of the conventional desktop device. Moreover, when the user changes the resolution, and one user is looking at a different view of the presentation with others, we also cannot expect that the pointed position means the same the actual position in SVG image as shown in Figure 3.4-(b) and (c).

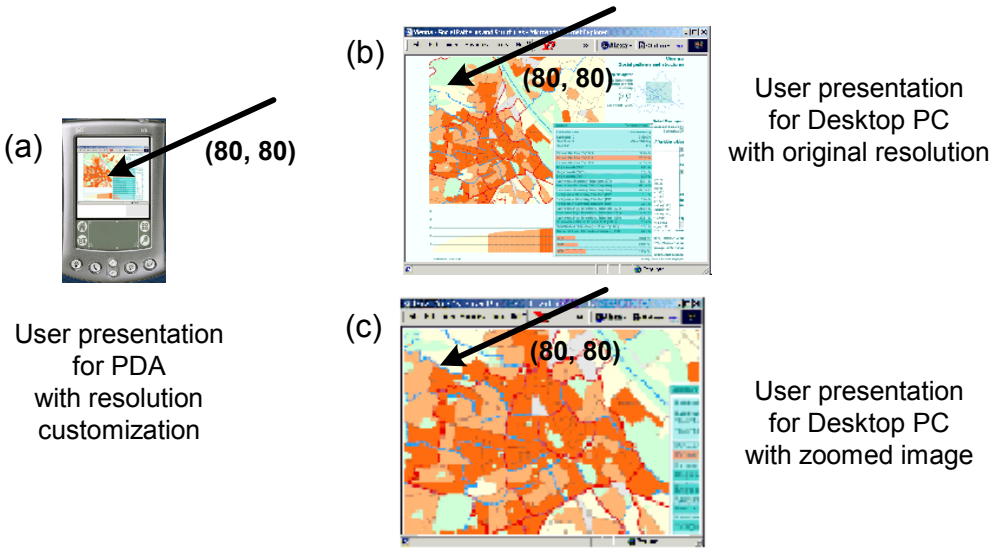


Figure 3.4 Event Transformation for Heterogeneous Customization of User Presentations

(a) Pointing the center of user presentation (80, 80) in the customized user presentation for the PDA

(b) Pointed user presentation for PCs with same position as (a) without filtering the event

(c) Pointed user presentation for PCs with same position as (a) without filtering event when the user zoomed the image.

Therefore, each stage should provide event filtering processing to map the event from the customized environment to the original environments. The most popular method is to keep track of history. For every pipeline, the activities are stored in a history buffer (a file). For simple processing, this strategy is very useful. In the SVG example, this can be used easily to recalculate the previous object based on the history.

However, there are some sophisticated tasks for which it complicated to get to the previous object. If the data is compressed to conserve network bandwidth, it needs specific reverse functions called “decompress” or “defrost” for accessing the previous object. Unfortunately, not all transcoding methods provide a reverse function. Since current transcoding methods, such as CCS or codecs, do not consider reverse functionality, this approach is rather difficult to implement. Therefore, we realized that

this is one of the major problems of the document object modeling in collaborative and interactive information systems.

In the SVG example, pointing at the (80, 80) event should be filtered to enable processing in the PCL stage. For instance, captured event (80, 80) is captured as the stylus clicking at the position (80, 80) on the PDA screen. This event is filtered to clicking (x, y) in an image displayed via the browser in the PCL stage. The event with position (x, y) is passed to the DTX stage and the DTX stage maps (x, y) to (x', y') which no longer corresponds to the PDA coordinates. Finally, the event arrives in the CTL stage with the actual coordinate (x', y') of the original image. The CTL stage filters the event to a semantically meaningful command, such as `svgTest.addelement(arrow, x', y')`.

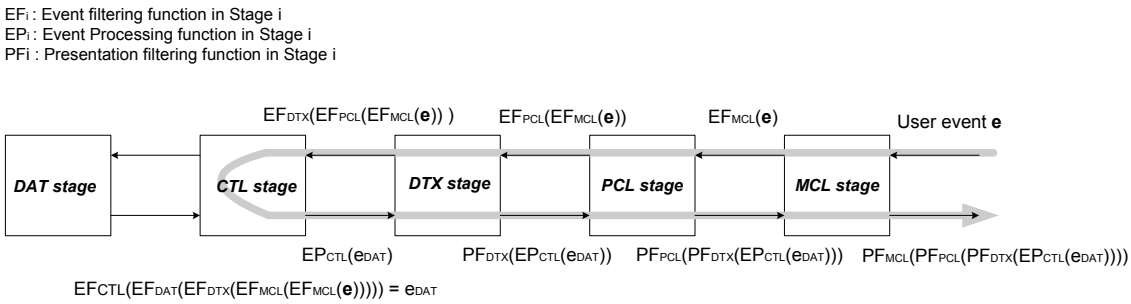


Figure 3.5 Data flow of SVG example

The event `svgTest.addelement(arrow, x', y')` is processed by event processing in the CTL stage, and the user presentation is generated as an output of the event processing. The output from the CTL stage is passed to the DTX stage for filtering the user presentation. Also the output of the DTX stage is delivered to the PCL stage, filtered and passed to the MCL stage. Finally, the MCL stage provides the actual user display depicted with the bitmap information. Figure 3.5 shows the data flow for the steps involved in the SVG event example.

We define the functions, event filtering, event processing, and presentation filtering for stage I, as EF_i, EP_i, PF_i. The result of each function performing the event processing feeds each other as input or output. Figure 3.5 illustrates how each function passes its output as input for the next stage in the dataflow.

As seen in the example, this event processing does not traverse all the stages of MDPA. User events in MDPA may or may not traverse all the stages in MDPA. We classify existing events with this traversing pattern, and describe it in section 3.2.2.

3.2.2 Event Processing Patterns in MDPA

A distributed service supports various types of events. We classify the existing events based on the patterns of event processing in MDPA. The user event in MDPA is processed in three major steps: event filtering, event processing, and presentation filtering. Each stage contains every component based on the characteristics of the stage. However, not every user event requires the execution of all the components, or even traversing all the MDPA stages. As an extreme example, there is a user event processed only in the MCL stage, such as moving the mouse. The display just shows the new position of the mouse pointer. It does not perform any function other than repositioning the mouse pointer in the user's display. This event only requires event processing in the MCL stage. If each event should traverse every stage of the MDPA, including mouse pointer repositioning events, obviously the response times for certain user events will be unnecessarily high. Nevertheless, we still need full stages of MDPA for a service. It is because a service contains various types of events which require visiting stages in different patterns.

We classify the types of user events according to the traversal pattern in the path for event and presentation: MLC event, PCL event, DTX event, CTL event, and DAT event.

As we mentioned in this section, there are events that are only processed in the MCL stage such as mouse moving, keyboard typing and echoing the typed character, etc. We classify these events as MCL events. MCL events do not traverse to other stages except for MCL stage. The output of event processing is directly delivered to the user display.

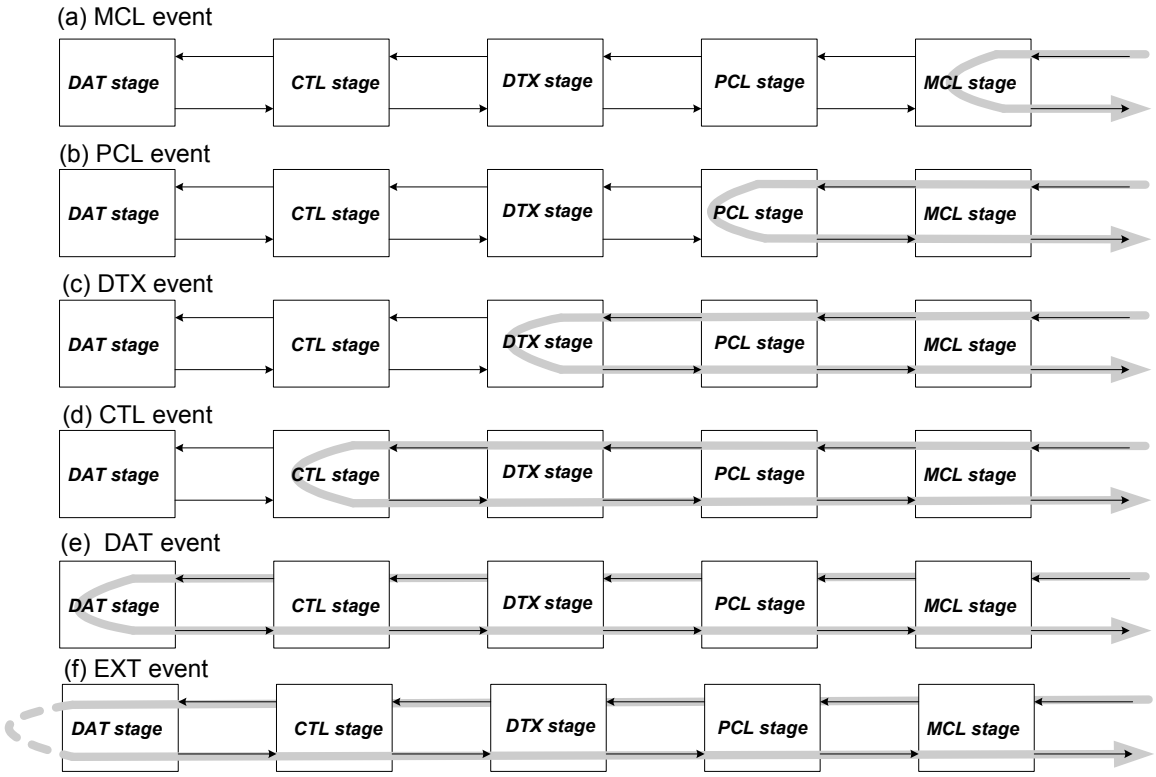


Figure 3.6 Dataflows in Different Event types

There are some user events, which, although similar to MCL events, but actually be should classified carefully in different groups, such as clicking a button or a turning dial. In user interfaces, we click a mouse at different locations, but if the position is defined as an area specifying a “button”, we see the button area indicating a button-press or if it is dial, the dragging mouse event also acts as turning the dial. Without defining the button or dial in the PCL stage, a user cannot receive the display. We classify these events as PCL events and this type of event should traverse both the MCL and PCL stages. For instance, the event “clicking position (x, y)” is passed to PCL stage, and it is filtered to “push button A”. The event “push button A” is processed in PCL stage, and generates a new presentation depicting a button press in the relevant area. The new presentation is actually presented as bitmap information and changes the information in a framebuffer.

Compared to distinguishing MCL and PCL events, it is more straightforward to classify DTX, CTL, and DAT events. The events regarding the transformation of presentations are classified as DTX events, which include events such as changing

resolution, or changing image format. These types of events are captured in MCL and transmitted to PCL. Filtered events in MCL and PCL stages are fed into the DTX stage as an input. The DTX stage filters the event and processes the event to generate a new presentation view. The newly generated presentation is passed to the PCL stage along the "presentation generating" path and the presentation filter in PCL processes the new presentation to fit the user's display. Finally, the MCL receives the output of user presentation from the PCL stage, and displays bitmap image data on to user's device. CTL events pertain to those events that require access to parsed data objects located in the CTL stage. The example of a SVG event demonstrates this type of event in section 3.2.1. DAT events deal with access to resources such as loading a new document or refreshing a page. Meanwhile, the EXT event is a classification for events that require access to resources located outside the pipeline. Loading documents from a Web server, sending a query to DBMS, or accessing a remote file server are examples of EXT events. Figure 3.6 presents the patterns of dataflow for each event type. Now we describe the detailed structure of the stages in 3.3.

3.3 Architecture of a Stage

Each stage of MDPA includes major components that comprise: interface, presentation filters, event filters, event processor, and stage manager. Figure 3.7 depicts the architecture of a stage while depicting the dataflow in the SVG browser example in section 3.2.1.

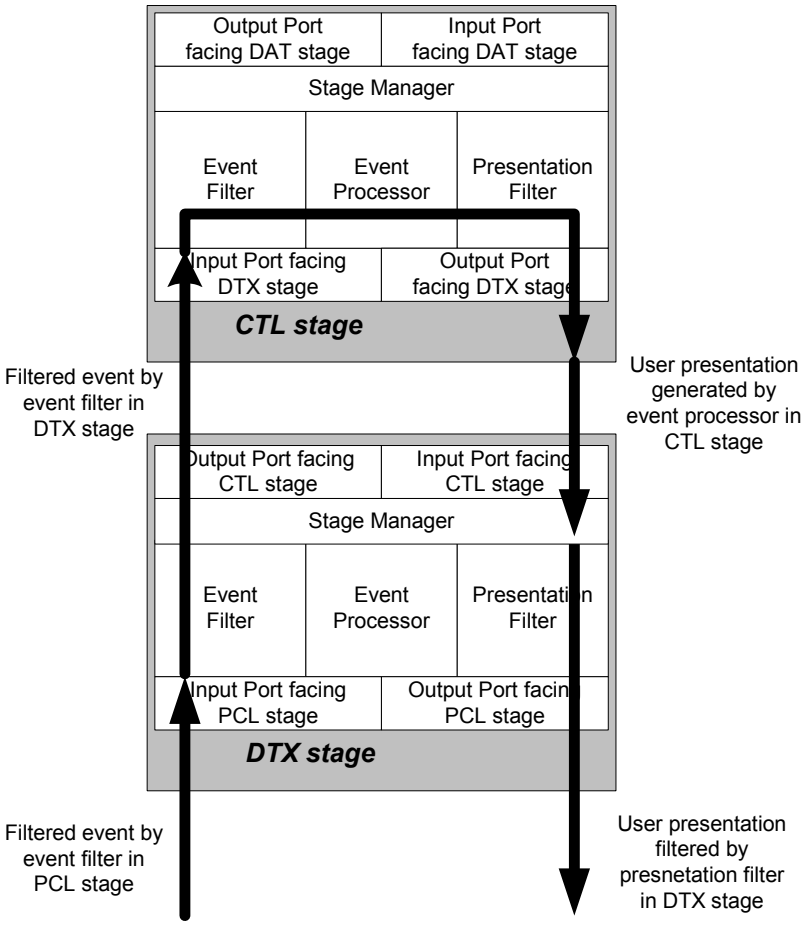


Figure 3.7 Architecture of a Stage in MDPA with dataflow of CTL event

- Interface: The interface in MDPA is a set of ports designed to face other stages, users, or resources. We distinguish these ports as input and output ports according to the direction of data transmission.
- Presentation Filters: To allow for heterogeneous devices, MDPA is designed for customizing user presentation views. Each stage processes the filtering of user presentation based on the characteristics of user devices and the functionality of the service.
- Event Filters: To process events while adapting to heterogeneous devices, each stage of MDPA provides event filters. A stage has one or multiple event filters, and also may have no filter. Another purpose of event filtering is regenerating an event for the stage. For instance, the event captured in the

MCL stage cannot be processed in the CTL stage without reformatting. Since each stage processes its task in a modular way, the event reformatting also should be performed individually in each stage.

- Event Processor: Each stage contains an event processing component.
- Stage Manager: Each stage provides stage manager to support efficient data flow, such as data caching, event queuing, etc. A stage manager also provides concurrency handling.

Figure 3.7 also shows partial data flow of a CTL event whose actual processing is performed in the CTL stage. The input port facing the DTX stage receives a filtered event through MCL, PCL, DTX stages. The event filter in the CTL stage reformats the received event to a DOM interface function for accessing the DOM tree in the CTL stage. Now actual accessing of the DOM tree is executed in the event processor of the CTL stage. The modified DOM, which is the output of the Presentation filter of the CTL stage, is passed to the DTX stage via output port facing the DTX stage. The DTX stage styles the document, makes adjustments based on the user profile, and passes the output to the PCL stage. After the traversed of the presentation through the PCL, and MCL stages, the users browse the new SVG image in their browser.

As shown in the SVG example, not every event processor performs their task. Meanwhile, the direction of dataflow is reversed at the stage that the event processor performs its task in.

The stage manager maintains the instance of the data pipeline. Since the data pipeline is structured with linked stages, if the stage is failed, the pipeline is broken. We assume that the MDPA data pipeline processes its event only when all the stage is alive. The stage manager handles the object instance of a stage.

3.4 Modularization and Packaging

In MDPA, flexibility is considered to support various types of services or deploy to infrastructure. To support various characteristics of functionalities and to adapt to different software infrastructures, the data pipeline should be able to define the data

processing with appropriate granularity. The flexible packaging of stages also makes adapting new infrastructure possible, such as Web services or P2P infrastructures.

To satisfy the need for flexibility, MDPA imposes three major requirements for each stage: modularization of stages, accessibility to resources and communication between stages.

First, every stage of the MDPA is designed as an independent module. The functionality of a stage is independent of the other stages. After a stage takes an output object from a prior stage as input, the stage is able to generate its output without invoking any process located in the other stages. Recursive processing over stages are not allowed in MDPA.

Second, the stages of MDPA need to provide an accessing method to access remote or local resources. When a function within a stage needs to access other resources, such as user profiles or stylesheets, it is able to access the resource based on capabilities available in the stage. Otherwise, the stages would be severely limited when they are packaged for various functionalities.

Third, MDPA requires predefined communication ports between stages to structure a pipeline architecture. With the input and output ports, each stage can pass the object to adjacent stages. A careful definition of input and output port is important to the design of a collaborative service, where the input or output ports are shared, and master's input or output object is fed to other participant's stage as if it is from their own pipeline.

Section 3.5 shows how MDPA is deployed within a Web service infrastructure.

3.5 Web Service Approach to MDPA

The Web service concept allows objects to be distributed across web sites where clients can access them via the Internet. Moreover, the use of XML [Bray+00] and standard interfaces like WSDL (Web Services Definition Language)[Christensen+01] or UDDI (Universal Description, Discovery and Integration of Web services)[UDDI+00] provides interoperability between services, including Grids[Grid], which comprise robust largely asynchronous shared resources. The current Web service infrastructures, such as Apache project's Jetspeed [Jetspeed] or WebSpheres [WebSphere] from IBM, provide

services and development tools for mobile devices. However, these approaches based on portal services are not efficient to support real time services such as synchronous collaborative services.

We addressed the emerging heterogeneous user environment comprising PDAs (Personal Digital Assistants) and generalize it to universal accessibility. Mobile systems are typically slow, unreliable, and have unpredictable temporal characteristics. Further, the user interface is clearly limited. The design of distributed mobile applications needs to identify the practicalities, reliability, and possibilities of continuous interaction and integrate synchronous and asynchronous collaboration. One of the steps to enable universal accessibility is by intelligently defining user "profile" and the semantic of Web services.

3.5.1 Dataflow in Web service

An object in Web services is typically in some pipeline, as seen in Figure 3.8, somewhere between the original object to the displayed user interface. Each stage of the pipeline is a Web service with data flowing from one stage to another. Rather than a simple pipeline, one can have a complex dynamic graph linking services together. In this section, we deploy our MDPA pipelining architecture. The modular design of MDPA enables the Web service to be designed based on the characteristics of the processing. Therefore the reuse of a given Web service for other distributed services is possible. We consider the output stage of MDPA as a "document" – each with its own document object model; preferably different instances of the W3C DOM (Document Object Model). The final user interface could be a pure audio rendering for a visually challenged user, or a bitmap transmitted to a primitive client not able to perform full browser functions.

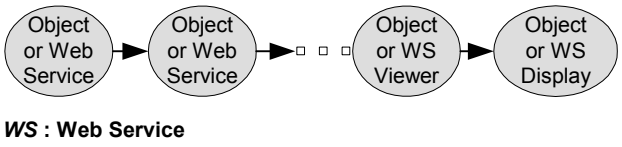


Figure 3.8 Dataflow in Web service

3.5.2 Deployment of MDPA within Web service infrastructure

We have presented the functionalities of each stage in MDPA. The stages in MDPA can be deployed as a Web service. Since MDPA is designed with the module, the functionality defined in any stage of MDPA can be deployed as a stand alone Web service.

We can deploy MDPA architecture into the Web service infrastructure by defining input/output ports with a standard interface, such as WSDL. Each stage of MDPA can be implemented in various ways. The communication between stages is processed via these ports, if each stage is located in different Web services.

Current multi-functional distributed services require a scalable framework to such as new Web services for adapting third-party Web applications. MDPA supports the integration of multiple services and the Web service approach provides the environment of aggregating with its standard such as UDDI or WSDL. Local or remote Web services are integrated into portals as portlets, which are user-facing, interactive web application components. A Web service is started with the exchange of *messages* between the service provider and the requestor. WSDL defines the exchange of messages between the service provider and the requestor as an *operation* [Christensen+01]. The messages in operations are described abstractly, and are bound to a concrete network protocol and a message format. The service is defined with an input or output port as a minimal description. Our research extends negotiation capabilities on output ports to support diverse dynamic client requirements for universal access.

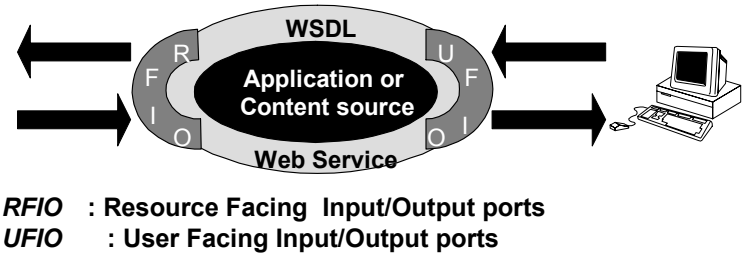


Figure 3.9 Input and Output Port Defined in Web service for MDPA

In order to describe a general approach to collaboration, we need to assume that every Web service has one or more ports in each of the four classes shown in Figure 3.9 . The first class is *resource-facing input ports*, which supply the information needed to define the state of the Web service. *User-facing input port*(s), which allow control information to be passed by the user, may augment these resource-facing input ports. The final class is *user-facing output ports* that supply information needed to construct the user interface. Asynchronous collaboration can share the data (e.g. URL for a Web page or body of an email message) needed to define a Web service (display Web page or browse e-mail in examples).

3.5.3 Packaging of Stages as Web service

The stages of MDPA can be packaged as Web service in various ways. Figure 3.10 depicts several of packaging styles. Figure 3.10-(a) packages the DAT, CTL and DTX stages as a Web service. In this packaging, each stage does not have to communicate with each other with the input/output ports defined following Web service semantics, such as WSDL. Since each stage does not provide Web service input and output ports, developers cannot reuse each stage for other services without knowing specific interface definition used inside of the Web service. Therefore, this type of packaging has low interoperability and reusability. However, we expect higher efficiency in this model.

Meanwhile, Figure 3.10-(b) shows the style where every stage is designed as individual Web services. Since every input and output port is defined with Web service semantics, each stage can be reused easily. Furthermore, interoperability with other services is obvious.

However, in real pervasive services, multiple filtering is processed in the DTX stage to generate a user presentation with multiple transcoding technologies. Transcoding is one of the most popular ways to tune content from a service provider. Transcoding is the transformation that converts the multimedia object from one form to another, frequently trading object fidelity for size, and is used to convert image or video formats (reduction resolution or compress data). Moreover, it is more efficient to develop some very expensive rarely used transcoding technologies as individual Web services. Otherwise, services are designed do not use the resources efficiently. Figure 3.10 illustrates how

multiple Web services are collapsed into a single stage. This style provides the interoperability and reusability to deploy MDPA within a Web service infrastructure. Additionally, it provides a more flexible design method for designing services with MDPA.

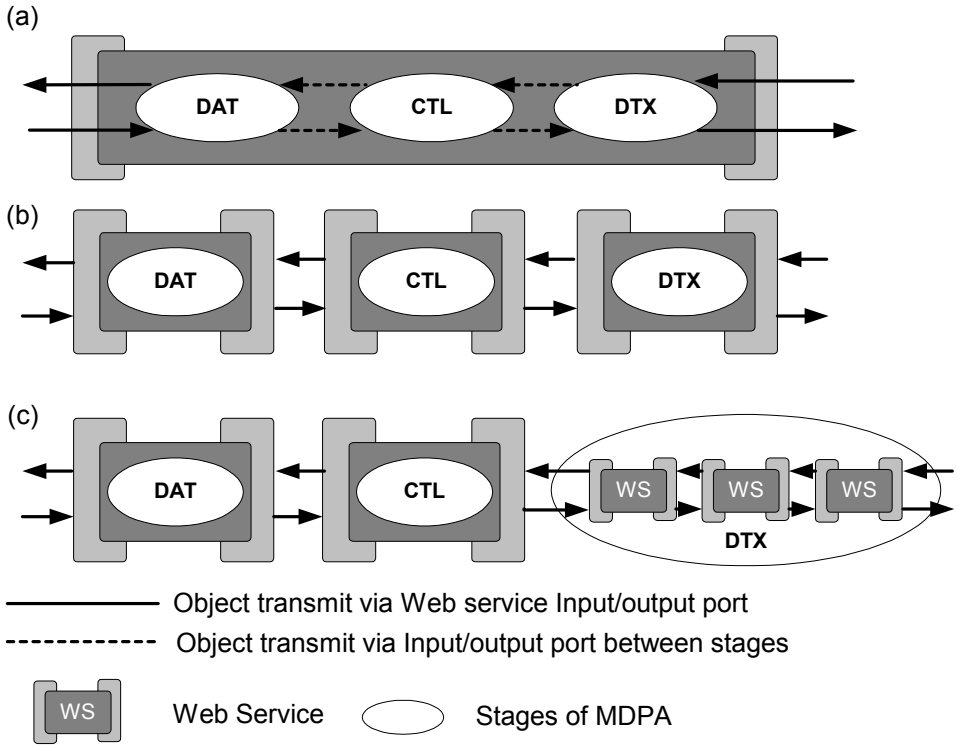


Figure 3.10 Various Styles of Packaging Stages with Web service infrastructure

3.6 Summary

In this chapter, we introduced MDPA which is an event based software architecture designed for heterogeneous user environments. We presented the overview of MDPA in section 3.1 with a brief description of its stages and functionality. In section 3.2, we described the event processing in MDPA. Based on the event processing, the basic architecture of stage was described in section 3.3. The MDPA contains interface, event filters, presentation filters, event processor, and stage manager as major components. The

interface of MDPA is a set of input and output ports for communicating with other stages. Each stage communicates with other stages, resources, or users via these ports.

Structuring a service with MDPA requires appropriate design for modularity of stages and the packaging of stages. We modularize this process with functional characteristics and approached flexibility with various packaging styles.

The Web service infrastructure provides interoperability using XML and standard interfaces like WSDL or UDDI. We explained how we can deploy MDPA in a Web service infrastructure by defining input and output ports with WSDL. We also discussed the design issues to designing MDPA as Web service.

In chapter 4, we will describe the modeling of MDPA and each stage of MDPA based on the discussion of this chapter.

CHAPTER 4 MODELING MDPA

The MDPA is modeled with the software architecture pattern of *Pipes and Filters* to show its dataflow clearly. Originally, the pattern of Pipes and Filters is structured with a set of filters [Schach+99], which denote the functions, and set of pipes which link these filters. We define a set of filters and pipes as a stage which processes its task modularly. Since MDPA is event based software architecture, a service in MDPA is specified with the event sets that are supported in the service. The processing of an object in a distributed service is refined with five stages in MDPA. Based on this refinement, we reclassify existing events according to the modularization process in MDPA. Therefore, an event in MDPA is also influenced by the characteristics of data processing. For example, an event related to only the current user display and an event processed with the original data object will be classified into different event types. We present the classification of events in MDPA, and the formal specification of the event system.

In this chapter we present the modeling of a service with MDPA. To model the service in MDPA, we start with the design of MDPA from presenting data flow in each stage. The representation of dataflow utilizes Petri net theory [Peterson+81] which is formalism for modeling the dataflow and concurrent event processing, with defining of its places and transitions therefore called as place and transition net (PT-net). We introduce the basics of PT-net and Modular PT-net [Christensen+00] in section 4.1.2.

The service designed with PT-net is analyzed with two methods. First, we simulate the data flow with a PT-net based simulator. Second, the reachability approach is used for verifying the connectivity. A description of each stage and the result of analysis are attached in Appendix A.

This chapter is organized as follows, section 4.1 presents the background of the modeling method, based on pipes and filters. Furthermore, we introduce PT-net and present the basic formal definitions. Moreover, we introduce higher level PT-net model, Modular PT-net. Section 4.2 describes the event model in MDPA. The classification of existing event sets and formal specification based on the event model is presented. Section 4.3 shows how we define each stage of MDPA, and finally a service of MDPA with Modular PT-net model. Section 4.4 mentions the analysis of defined service in MDPA, and presents how it can be verified based on the analysis. A summary of this chapter is followed in section 4.5.

4.1 Modeling MDPA

MDPA is modeled with the pattern of the Pipes and Filters to present the data flow in P2P Grid services. Compared to other architectural patterns such as client-server or object-oriented patterns, the pattern of Pipes and Filters enables the design to be more flexible with easier creation of prototypes [Buschmann]. In section4.1.1, we describe the advantages we can inherit when we adapt the Pipes and Filters pattern for the software architecture of P2P Gird in detail.

As a specification and design method, we utilize PT-net which is a widely known formalism for modeling dataflow and concurrent event processing. Presented for the first time by C.A. Petri in 1962 [Petterson+81] it has since been the focus of a lot of research work. The main uses of Petri nets are for discrete event simulation and modeling complex systems such as communication protocols or distributed databases. The major advantages of using PT-net in such areas are their expressiveness with regards to concurrency related concerns, their graphical representation, their mathematical foundation, and their excitability. PT-net is often ranked amongst the state-based formalisms, which may be due to a hasty assimilation with finite stage automata.

PT net models a system by a set of places and a set of operations (called transitions). The state of a system is modeled at any moment by a distribution of tokens in the net's place. Place and transitions are connected by directed arcs, which define when each transition is allowed to occur, and what the effect of its occurrence will be. A transition is allowed to occur when each of its input places holds at least one token. The occurrence of the transition consumes a token in each input place and sets a token; the occurrence of the transition consumes a token in each input place and sets a token in each output place. Moreover, an integer weight n is associated with each arc by the forward and backward incidence function, thus allowing the consumption or setting of n tokens in a place at a time. Generally, a transition occurrence may consume or produce tokens. Especially, we illustrate this architecture as *Modular PT-net model* [Christensen+00] to illustrate our modular architecture efficiently. Modular PT-net model illustrates a PT-net in which the individual modules, defined as PT-nets, interact via *shared places and transitions*. The Modular PT-net technology is useful for defining data pipelines that contain multiple modules with its modularity. Moreover, the concept of shared nodes is clearly applied to describe the packaging of modules.

In section 4.1.2 and 4.1.3, we will describe the basic formal definitions that we will use often in this dissertation to specify and design MDPA.

4.1.1 Architectural Pattern of "Pipes and Filters"

The major methods of describing the architecture of a software product might be object-oriented, pipes and filters, or client-server with a central server providing file storage and computing facilities for a network of client computer [Schach+99].

The architectural pattern which is "Pipes and Filters" is defined in [Buschmann+96] as,

> *"The* Pipes and Filters *architectural pattern provides a structure for systems that process a stream of data. Each processing step is encapsulated in a filter component. Data [are] passed through pipes between adjacent filters. Recombining filters allows you to build families of related filters."*

The filters are the processing units of the pipeline in classical pipes and filters pattern, and the pipes are the connectors—between a data source and the first filter, between filters, and between the last filter and a data sink.

For designing software architecture with the pattern of pipes and filters, we should divide the task into a sequence of processing steps. Next step is enabling the filters to execute concurrently. Finally, we should connect the input to the sequence of processing steps to some data source, such as an object. Also, we must connect the output of the processing sequence to some data sink, such as an object or display device.

A filter usually may be active or passive. An active filter runs as a separate process or thread; it actively pulls data from the input data stream and pushes the transformed data onto the output data stream. A passive filter is activated when an output from the filter is pulled or input to the filter pushed by its definition.

With using Pipes and Filters pattern, we can inherit the benefits [Buschmann+96] such as,

- *Intermediate files unnecessary, but possible.* File system clutter is avoided and concurrent execution is made possible.
- *Flexibility by filter exchange.* It is easy to exchange one filter element for another with the same interfaces and functionality.
- *Flexibility by recombination.* It is not difficult to reconfigure a pipeline to include new filters or perhaps to use the same filters in a different sequence.
- *Reuse of filter elements.* The ease of filter recombination encourages filter reuse. Small, active filter elements are normally easy to reuse if the environment makes them easy to connect.
- *Rapid prototyping of pipelines.* Flexibility of exchange and recombination and ease of reuse enables the rapid creation of prototype systems.
- *Efficiency by parallel processing.* Since active filters run in separate processes or threads, pipes-and-filters systems can take advantage of a multiprocessor.

A service with MDPA is designed as a set of modules in data pipeline. Each module is distinguished with its functional characteristics. A module is illustrated as a set of filters and pipes in MDPA. We define the modules as "*stages*". Therefore, the stage contains a set of filters and pipes.

4.1.2 Representation with PT-nets

A PT-net is composed of four parts: a set of places P, a set of transitions T, a weight function W, and a mapping function for the initial marking M_0. We start by giving the definition of PT-nets, and introducing the notations to be used in this chapter. We use the following definition of PT-net,

DEFINITION 4.1 *A* PT-net *is a tuple PN = (P,T,W,M_0), satisfying*,

(i) *P is a finite set of places*;

(ii) *T is a finite set of transitions. The sets of net elements are disjoint:* $T \cap P = \{\}$;

(iii)*W is the arc weight function mapping from* $(P \times T) \cup (T \times P)$ *into* N;

(iv)M_0 *is the initial marking.* M_0 *is a function mapping from P into* N.

Now we define markings and steps for PT-nets.

DEFINITION 4.2 *A* marking *is a function M mapping from P into* N *while a step is a non-empty and finite multi-set over T. The sets of all markings and steps are denoted by M and Y, respectively*.

We denote the set of multi-sets over a set A by A_{MS}.

The enabling and occurrence rules of a PT-net can now be defined.

DEFINITION 4.3 *A* step Y *is enabled in a marking M, denoted by* $M[Y\rangle$, *iff the following property is satisfied:*

$$\forall p \in P : M_1(p) = \left(M_1(p) - \sum_{t \in Y} W(p,t) \right) + \sum_{t \in Y} W(t,p) \ .$$

Note the summations above are over a multi-set Y. This means that *W(p,t) and W(t,p)* appears in Y. We say that M2 is directly reachable from M1 by the occurrence of step Y, which is denoted by: $M_0\ M_1[Y\rangle M_2$. $[M\rangle$ denotes the set of markings reachable from M.

Meanwhile, we describe transition with input and output functions also,

DEFINITION 4.4 *For a transition T, input and output* function *are defined as*,

I: T → P^∞ is the input function, a mapping from transitions to bags of places.

O: T → P^∞ is the output function, a mapping from transitions to bags of places.

For example, the transition in PT-net G of Figure 4.1 has input and output function, *I(t) = {p1},* and *O(t) = {p2}*.

Most theoretical work on PT-net is based on the formal definition of PT-net structures given above. However, a graphical representation of a PT-net structure is much more useful for illustrating the concepts of PT-net theory. A PT-net graph is a representation of a PT-net structure as a bipartite directed multigraph.

A PT-net structure consists of places and transitions. Corresponding to these, a PT-net graph has two types of nodes. A circle O represents a place; a bar | represents a transition. Since the circles represent places, we call the circles places. Similarly, we call the bars transitions.

Directed arcs (arrows) connect the places and the transitions, with some arcs directed from the places to the transitions and other arcs directed from transitions to places. An arc directed from a place *p* to a transition *t* defines the place to be an input of the transition. Multiple inputs to a transition are indicated by multiple arcs from the input places to the transition. An output place is indicated by an arc from the transition to the place. Again, multiple outputs are represented by multiple arcs.

A PT-net is a multigraph, since it allows multiple arcs from one node of the graph to another. In addition, since the arcs are directed, it is a directed multigraph. Since the

nodes of the graph can be partitioned into two sets (places and transitions), such that each arc is directed from an element of one set (place or transition) to an element of the other set (transition or place), it is a bipartite directed multigraph.

We utilize the graphical representation to present our design with PT-net to illustrate each stage. Meanwhile, we adapt a higher level concept of PT-net, which is Modular PT-net to represent whole architecture of MDPA in a more simplified fashion.

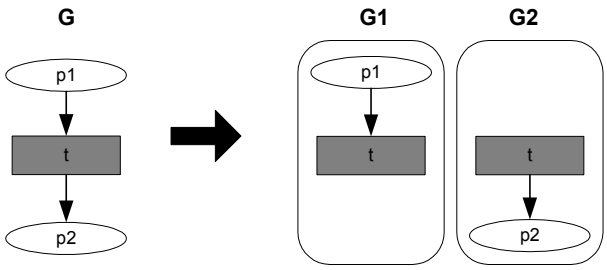


Figure 4.1Two modules Sharing Transition

4.1.3 Definition of Modular PT-nets

The use of high-level PT-net formalisms has enabled creation of PT net based models of large systems. We illustrate our system by means of Modular PT-nets in which the individual modules interact via shared places and shared transitions. Sharing is often accomplished using place fusion sets and transition fusion sets. These two strategies for communications are present in a number of models, e.g. see [Battiston+91,Christensen+94] for models using shared transitions and see [Huber+90, Jensen+92] for models using place fusion. Figure 4.1 shows the original PT-net G is modulated to G1 and G2 which are sharing transition t.

We will now give the definition of a modular PT-net.

DEFINITION4.5 *A* modular PT-net *is a triple MN = (S, PF, TF), satisfying the following requirements:*

(i) *S is a finite set of modules such that*:

(a) *Each module*, $s \in S$, *is a PT-net*:

$s = (P_s, T_s, W_s, Mo_s);$

(b) *the sets of nodes corresponding to different modules are pair-wise disjoint:*

$$\forall s_1, s_2 \in S : \left[s_1 \neq s_2 \Rightarrow (Ps_1 \cup Ts_1) \cap (Ps_2 \cup Ts_2) = \{\} \right].$$

(ii) $PF \subseteq 2^p$ *is a finite set of place fusion sets such that:*

(a) $P = \bigcup_{s \in S} Ps$ *is the set of all places of all modules*;

(b) *For nodes* $x \in P \cup T$ *we use* $S(x)$ *to denote the module to which x belongs. For all p in P we define* $M_0(p) = M_{0s(p)}(p)$;

(c) *members of a place fusion set have identical initial markings*:

$$\forall pf \in PF : \forall p_1, p_2 \in pf : \left[M_0(p_1) = M_0(p_2) \right]$$

(iii) $TF \subseteq 2^T$ *is a finite set of transition fusion sets where*:

$T = \bigcup_{s \in S} Ts$ *is the set of all transitions of all modules*.

For the sake of simplicity, we use the same names for objects (places or transitions) belonging to different modules, but which have to be fused together.

A Modular PT-net contains a finite set of modules, each of them being PT-net. These modules must have disjoint sets of nodes. Each place fusion set is a set of places to be fused together. 2^p denotes the set of all subsets of places. We mandate that all elements of a place fusion set have the same initial marking. Note that we do not demand the place fusion sets to be disjoint. Each transition fusion set is a set of transitions to be fused together. Similarly, we do not mandate that the transition fusion sets to be disjoint. Figure 4.1 presents Modular PT-net with sharing transition.

Stages in MDPA are represented as modules of a Modular PT-net. There are five modules in MDPA: DAT, CTL, DTX, PCL, and MCL. Each subset is defined as PT-net with a quintuplet <*subset.P, subset.T, subset.W subset.M*>. Each subset defines transitions for input and output to provide interface to other stages, resources, or users. Each transition located in a subnet represents the process performed in the stage.

With using Modular PT-net model, we define the design of service in MDPA. Prior to describe our design, we present the specification of event processing in MDPA.

4.2 Event Processing in MDPA

Data processing in MDPA is based on user events. We classify existing user event for the distributed services according to the pattern of event processing over the MDPA architecture. There are six categories of events: *MCL event, PCL event, DTX event, CTL event, DAT event,* and *External event*. Here the External event is defined as the event, which accesses the resource located in outside of pipeline such as loading Web Pages, or accessing remote DBMS.

A helpful starting point is the classification of user events in MDPA (see Figure 3.5). In the collaborative service, there are collaborative and non-collaborative events. If the event changes only the state of the service which captured event originally, the event is not considered as collaborative event. Meanwhile, in the collaborative service, master's event affects the state of a user's service. In this chapter we describe how MDPA supports non-collaborative event processing to show the basic mechanism of event processing. In chapter 5 we will focus on collaborative event processing in detail.

For demonstrative purpose, we present an example event set of SVG (Scalable Vector Graphics) browser. SVG is a 2D vector graphics standard format from W3C and has a structured XML syntax. In this example, we provide examples for 6 types of user events.

Table 4.1 User Event types and the Stages

Event Type	Filtering Event	Event Processing	Filtering User Presentation
MCL event	MCL	MCL	MCL
PCL event	MCL, PCL	PCL	PCL, MCL
DTX event	MCL, PCL, DTX	DTX	DTX, PCL, MCL
CTL event	MCL, PCL, DTX, CTL	CTL	DAT, DTX, PCL, MCL
DAT event	MCL, PCL, DTX, CTL, DAT	DAT	CTL, DTX, PCL, MCL

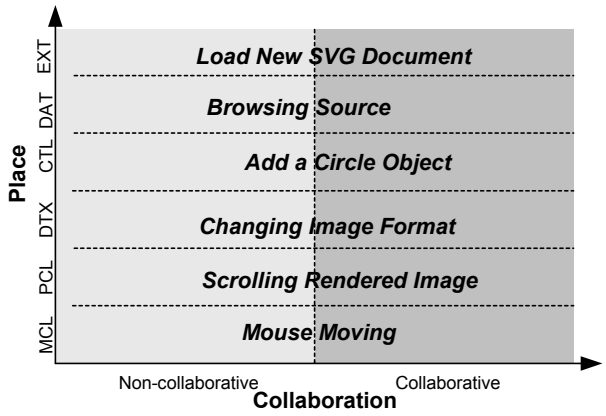


Figure 4.2 Classifications of events in MDPA

Event processing includes three steps of data processing: event filtering, event processing, and filtering of user presentation. As we described in chapter 3, the user event captured from a user device is filtered and transmitted to the stage where the event is finally executed. After the event is transmitted, the event is executed and starts to traverse the data pipeline in reverse direction to filter the user presentation. Table 4.1 illustrates the user event type and the stages which are visited for event processing in each step.

In current sophisticated services, the event usually performs multiple types of events to provide a more user friendly response: For instance, when the user wants to load a new document, current browsers perform EXT, PCL, and MCL events together. First, the user's mouse is moved to the area specified as a "loading" button and the event of clicking mouse once or twice is captured in MCL. Then, the clicking mouse is transferred to "push button" of GUI, because the area is specified as a GUI button, and the area is blinking, which is the event processing of PCL event. Finally, the event traverses pipeline, and loads new data. From the new data, the user's new presentation view is generated through the pipeline. This is EXT event type as we defined earlier in this section. This pertains to the user interface design. In this chapter, we do not consider the design of user interface. Every event here is a single activity related to user input.

4.3 Definition of a Service with MDPA

A service in MDPA is defined as a Modular PT-net. A set of modules denotes stages in MDPA. Each module is defined as a PT-net. Meanwhile, the modules are linked via set of ports which is called interface. Finally, a service is structured with the stages and interfaces as illustrated in Figure 4.3.

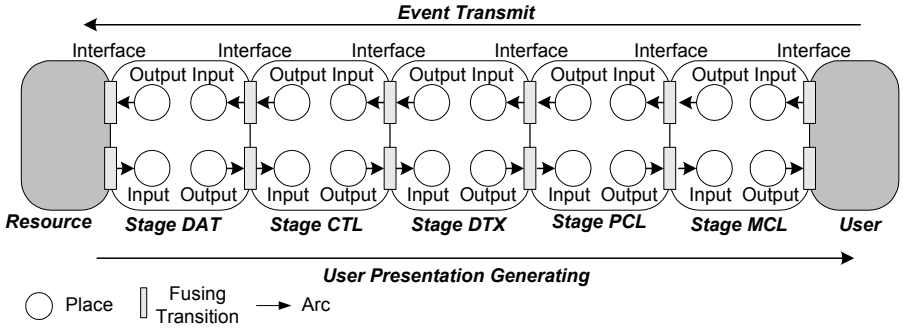


Figure 4.3 Stages and Interface of MDPA

In this section, first, we define the interface with the concept of fusing transitions. Definition of each stage follows. Finally, a service is defined with the stages and interface.

4.3.1 Interface of MDPA

The dataflow of MDPA is performed by communication via interfaces. Each interface contains a set of Input ports and a set of Output ports. Input and Output ports are denoted as a set of fusing transitions shared by communicating stages. Figure 4.4 shows how the transitions are shared and how stages are linked. We provide formal definition of interface in a stage of MDPA as follows,

DEFINITION 4.6 *The Interface of a stage s is defined as,*

$Interface_s$ = $Input\ port_s \cup Output\ port_{s,,}$ such that:

(i) *For node x, we use* $S(x)$ *to denote the stage to which x belongs.*

(ii) *Input port is a set of transitions t such that*,
$(S(O(t))) = s \wedge (S(I(t)) = s') \wedge (s \neq s')$,

(iii) *Output port is a set of transitions t such that*,
$(S(I(t))) = s \wedge (S(O(t)) = s') \wedge (s \neq s')$.

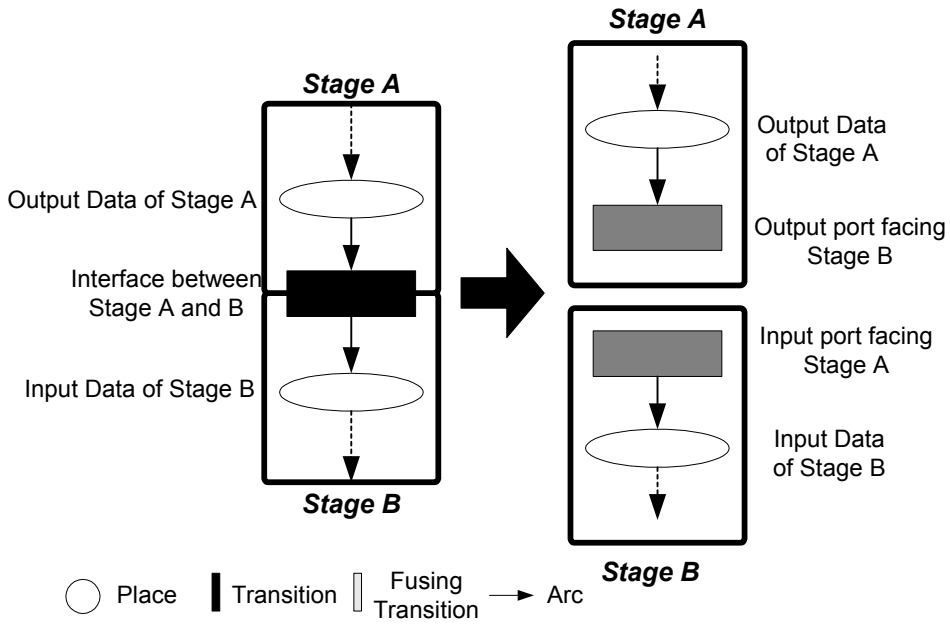


Figure 4.4 Defining Interface with Fusing Transitions

The interface links each stage and provides a communication method for resources or users. Therefore, the input port is defined as a set of transitions which has one arc from outside of the stage and another arc to inside of the stage. Similarly, the output port is defined as a set of transitions which has one arc from inside of the stage and another arc to outside of the stage. Furthermore, in the Modular PT-net, the interface is defined a set of fusing transitions, because each port is shared by two adjacent stages.

Since MDPA is designed for supporting service between resources and user based on P2P Grid environments, the pipeline should provide the interface for resources and user. The definition of interface is also available for resources and users.

4.3.2 Basic Stage Managements

Since the modular architecture is designed to process the data in each module independently, the object instance needs to be managed efficiently. Otherwise, it can cause unnecessary transitions between stages and redundant data processing in each stage. To avoid being prone to such disadvantages, we provide functionality of managing object instance to individual stages. These functionalities are processed in Stage Manager.

The basic object managing of MDPA is in the following way:

- *Instance of stage*

  We assume that the pipeline is structured only when every stage is available. If a stage is crashed, the pipeline is broken. The place which shows "This stage is available." is marked if the stage is available.

- *Concurrency*

  The input pertaining to presentation generating and event transmit path, along with multiple events transmits, can result in concurrency problems. We synchronize events and presentation view in every stage and maintain event queue to prevent transition conflicts or deadlocks.

- *Cache the Output*

  The output object at a given stage is accessible even after the processing of that stage is finished. This is for providing reusability to MDPA. By keeping the object instance, the output can be processed with the followed user events without reproducing. Each stage consists of a cache to store the "n" outputs of data processing.

We describe the stages from section 4.3.3 through section 4.3.7. Each PT-net is illustrated over the basic architecture of a stage depicted in Figure 4.5. There are three possible dataflows in a stage, and DAT stage has an additional dataflow to process the EXT event. As depicted in Figure 4.5, event is transmitted through input port, event filter, and output port. The presentation from adjacent stage is filtered via input port, presentation filter, and output port. Meanwhile, the event is processed in a stage through input port, event filter, event processor, presentation filter and output port. In this case, the stage processes the data in two opposite directions. Since the EXT event is required for accessing resources, dataflow points to outside the stage via its output port. The dataflow of EXT event processing is only for the DAT stage.

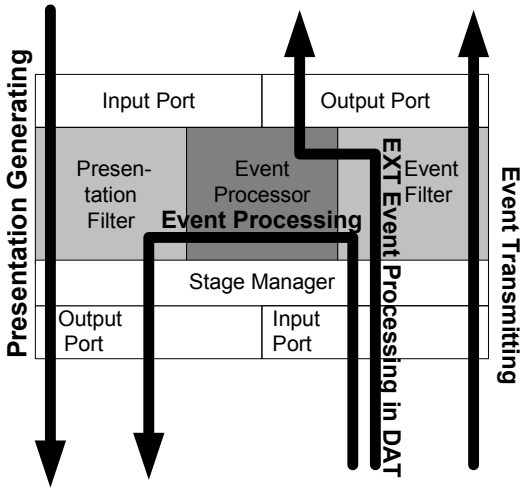


Figure 4.5 Possible dataflows in a stage

4.3.3 The MCL stage

The MCL stage is a stage of MDPA, which is closest to user devices. MCL stage defines input events from the user's input behavior directly, such as clicking a mouse button or moving a mouse etc. The user presentation view is actually rendered and accessed in the MCL stage for each user device. For the visual user interface, actual framebuffered image is updated with an MCL event. The event queue and cache are also provided here as stage manager. MCL stage contains input/output ports facing the users in the path of event transmit, and input/output ports facing the PCL stage in the path of presentation generating. We call these input and output ports as interface of MCL stage. Meanwhile, these ports are shared with facing stage or user. Therefore, we define the input and output ports as fusing transitions in Modular PT-net.

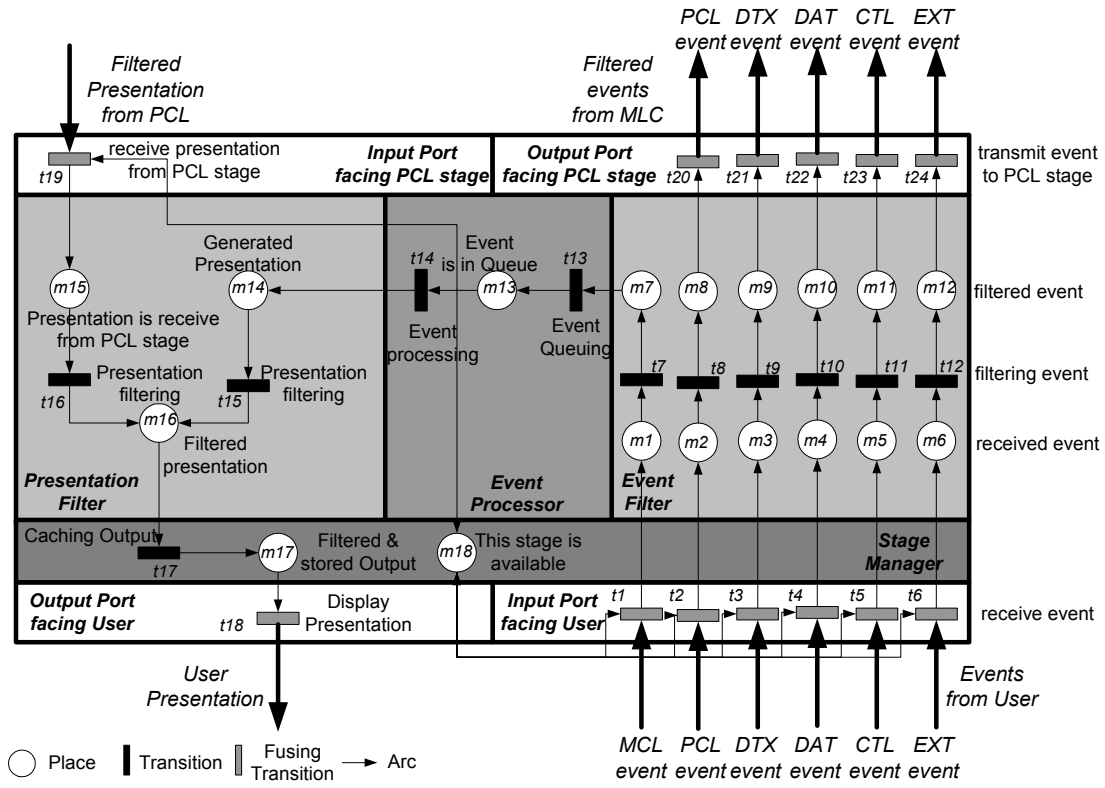


Figure 4.6 PT_{MCL} : MCL stage with PT-net

Therefore, MCL stage is denoted as a PT-net, such as,

DEFINITION 4.7 MCL stage *is defined as a PT-net such that:*

(i) $PT_{MCL} = (P_{MCL}, T_{MCL}, W_{MCL}, Mo_{MCL})$ *defined in Figure 3.5 graphically*,

P_{MCL}, T_{MCL} *are defined in* Figure 4.6 *graphically.*

$\forall p \in MCL, W_{MCL}(p) = 1$, *constantly.*

$Mo_{MCL}(mcl.m18) = 1$, *initially.*

(ii) $Interface_{MCL}$ = *Input port facing PCL stage*

$\cup$ *Input port facing USER*

$\cup$ *Output port facing PTL stage*

$\cup$ *Output port facing USER*, *such that*:

Input port facing PCL stage = {mcl.t15}

Input port facing USER = {mcl.t1, mcl.t2, mcl.t3, mcl.t4, mcl.t5, mcl.t6}

Output port facing PCL stage = {mcl.t20, mcl.t21, mcl.t22, mcl.t23, mcl.t24}

Output port facing USER = {mcl.t18}

(iii) *Each Input and Output functions for transitions are defined in Figure 4.6 graphically*.

The place *mcl.m18* which denotes "This stage is available" is required to be marked to access PCL stage or user, initially. This place checks whether the instance of stage is alive or not, while also providing synchronization for incoming events and presentations.

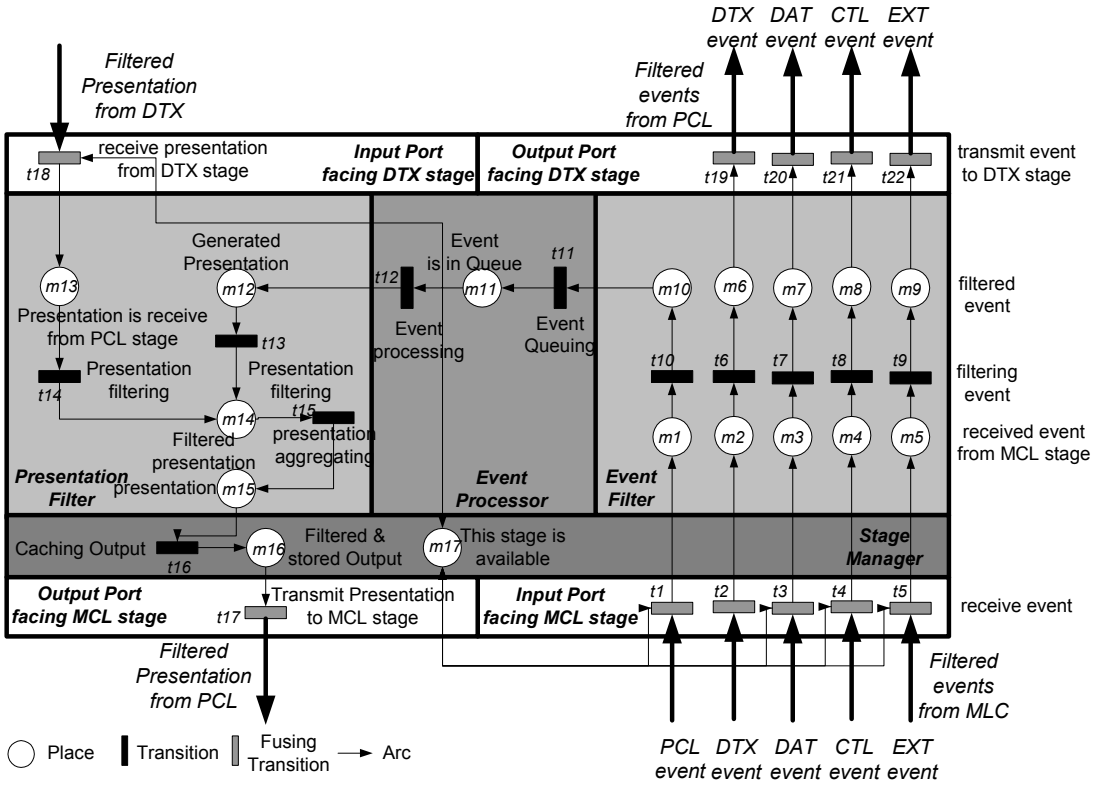


Figure 4.7 PT_{PCL} : PCL stage with PT-net

4.3.4 The PCL stage

Figure 4.7 illustrates the design of PCL stage, which provides input and output interfaces for the DTX stage and MCL stage respectively. PCL stage generates the actual user presentation. Besides event filtering and generating user presentation view for single MDPA, the PCL stage is designed specifically for aggregating multiple pipelines into a unified user presentation view. Therefore, the fusing transition *pcl.t18*, denoting "receive presentation from DTX stage" also supports the input from DTX stage of other data pipelines. We discuss the interoperating between data pipelines in chapter 5.

Similar to the MCL stage, there are cache and event queue and concurrent controlling units in the Stage Manager.

Therefore, we define PCL stage as,

DEFINITION 4.8 PCL stage *is defined as a PT-net such that:*

(i) $PT_{PCL} = (P_{PCL}, T_{PCL}, W_{PCL}, Mo_{PCL})$

P_{PCL}, T_{PCL} *are defined in* Figure 4.7 *graphically.*

$\forall p \in PCL$, $W_{PCL}(p) = 1$, *constantly.*

$Mo_{PCL}(pcl.m17) = 1$, *initially.*

(ii) $Interface_{PCL}$ = *Input port facing DTX stage*

$\cup$ *Input port facing MCL stage*

$\cup$ *Output port facing DTX stage*

$\cup$ *Output port facing MCL, such that*:

Input port facing DTX stage = {pcl.t18}

Input port facing MCL stage = {pcl.t1, pcl.t2, pcl.t3, pcl.t4, pcl.t5}

Output port facing DTX stage = {pcl.t19, pcl.t20, pcl.t21, pcl.t22}

Output port facing MCL stage = {pcl.t17}

(iii) *Each Input and Output functions for transitions are defined in Figure 4.7 graphically*.

4.3.5 The DTX stage

DTX is the stage designed for adapting data according to device profiles or user preferences. Therefore, there are various data transfers in the DTX stage. For complicated data adapting process, the presentation filtering can be multiple steps of filters. We discuss the data adapting process in chapter 6 in detail. Another important functionality of DTX is reverse filtering of events. DTX stage must transmit information in a way such that user interactions can be properly passed back from the user with the correct semantic meaning. Let us assume that DTX translated the coordinate of graphical data to manipulate the output of CTL. For an event that occurred in translated coordinate, DTX must map to the information according to the original coordinate. Otherwise, the user cannot expect the accurate result in their generated presentation.

However, the reverse mapping can be quite complicated and it is not clear how this is achieved in general as the pipeline. Current browsers and transformation tools (such as XSLT) do not appear to address this. For this reversibility problem, MDPA allows the generation of new output from the original raw data set in the DAT stage. The ambiguity of reverse functionalities still exists, but we can expect that every reverse function will get correct output with this design.

Adapting remote processing is also supported in DTX stage, such as transcoding technologies or accessing user profiles. Therefore, the deploying DTX can be in various styles. We describe the adaptability of MDPA in Chapter 5 in detail with its various types of deployments.

DEFINITION 4.9 DTX stage *is defined as a PT-net such that:*

(i) $PT_{DTX} = (P_{DTX}, T_{DTX}, W_{DTX}, Mo_{DTX})$

P_{DTX}, T_{DTX} *are defined in* Figure 4.8 *graphically.*

$\forall p \in DTX$, $W_{DTX}(p) = 1$, *constantly.*

$Mo_{DTX}(dtx.m14) = 1$, *initially.*

(ii) $Interface_{DTX}$ = *Input port facing CTL stage*

$\cup$ *Input port facing PCL stage*

$\bigcup$ *Output port facing CTL stage*

$\bigcup$ *Output port facing PCL*, *such that*:

Input port facing CTL stage = {dtx.t15}

Input port facing PCL stage = {dtx.t1, dtx.t2, dtx.t3, dtx.t4}

Output port facing CTL stage = {dtx.t16, dtx.t17, dtx.t18}

Output port facing PCL stage = {dtx.t14}

(iii) *Each Input and Output functions for transitions are defined in Figure 4.8 graphically*.

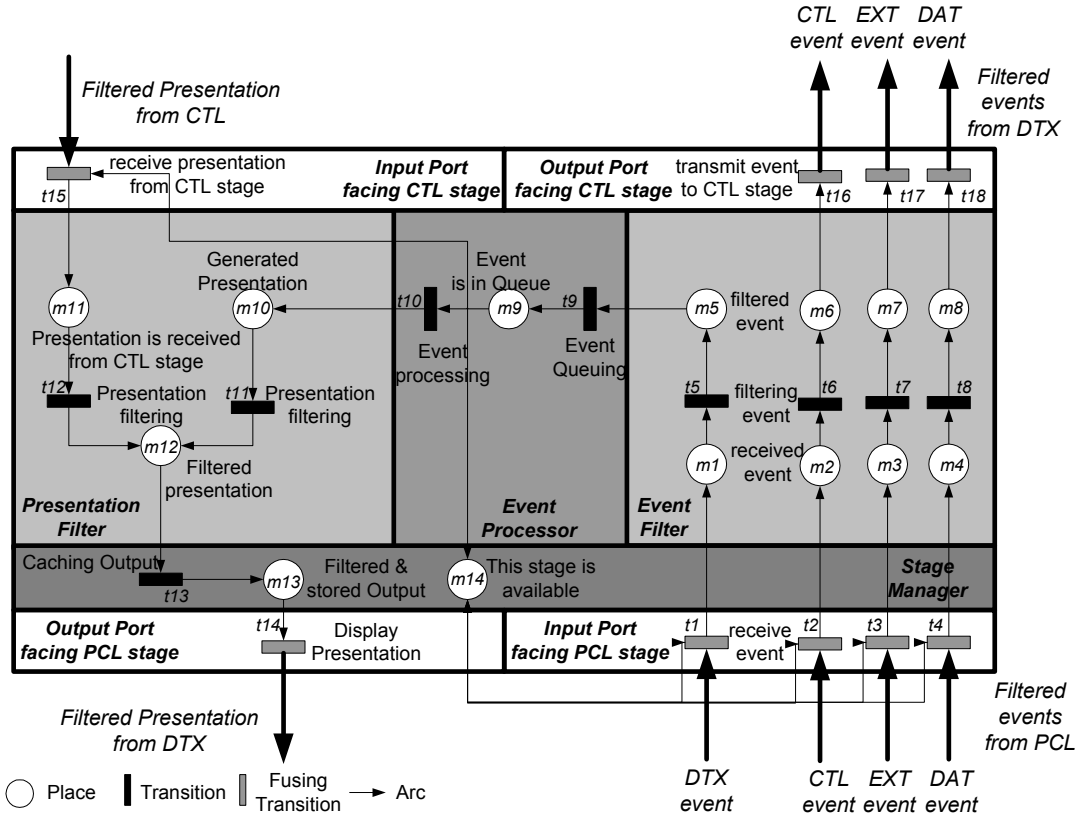


Figure 4.8 PT_{DTX} : DTX stage with PT-net

4.3.6 The CTL stage

CTL provides control logic to the data object from DAT stage. Therefore, the event processing in the CTL stage requires accessing original data object with the semantic meaning. For the document based object model, the output of the CTL stage is a DOM object. Via the DOM interface, the event in CTL accesses the data object. However, not all data processing is based on the DOM in actual distributed services. Therefore, the CTL stage also depends on the object structure. New presentation generated in CTL stage starts to traverse the pipeline towards the user display.

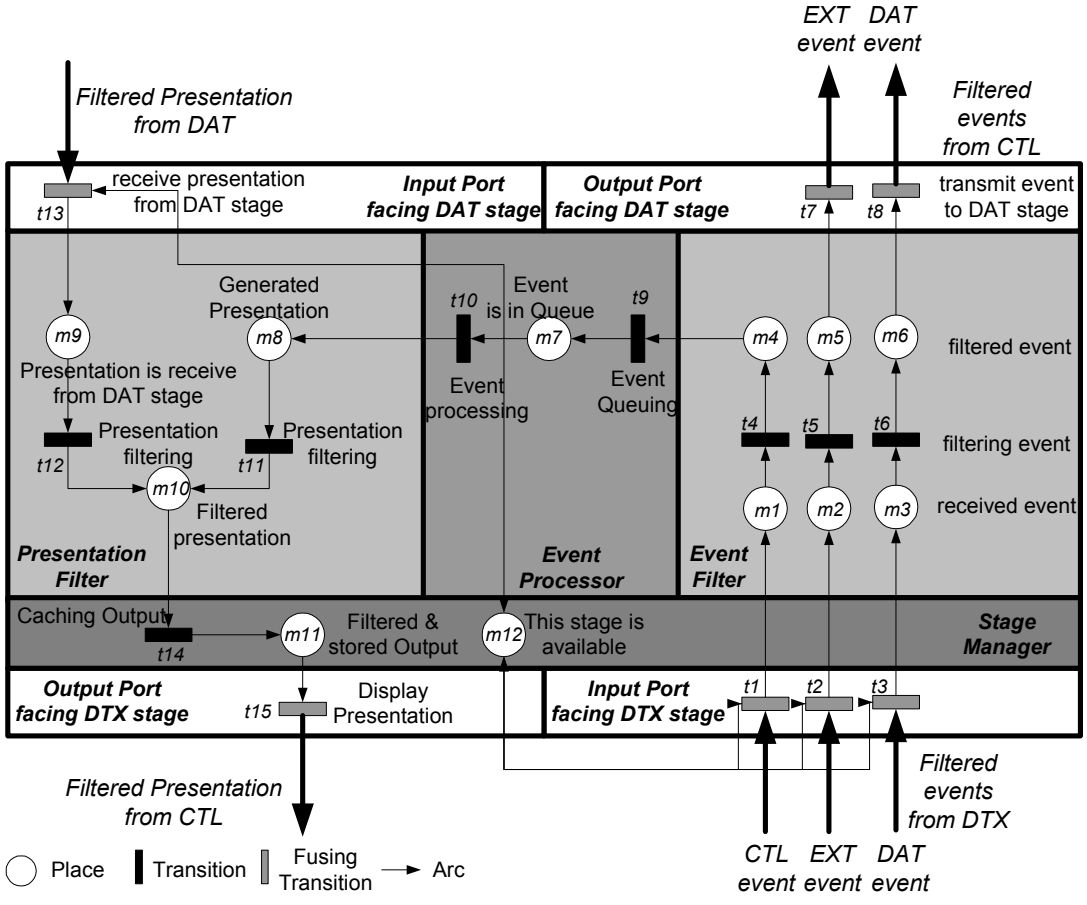


Figure 4.9 PT_{CTL} : CTL stage with PT-net

We define CTL stage as,

DEFINITION 4.10 CTL stage *is defined as a PT-net such that:*

(i) $PT_{CTL} = (P_{CTL}, T_{CTL}, W_{CTL}, Mo_{CTL})$

P_{CTL}, T_{CTL} *are defined in* Figure 4.9 *graphically.*

$\forall p \in CTL$, $W_{CTL}(p) = 1$, *constantly.*

Mo_{CTL}*(ctl.m12) = 1, initially.*

(ii) $Interface_{CTL}$ = *Input port facing DAT stage*

$\cup$ *Input port facing DTX stage*

$\cup$ *Output port facing DTX stage*

$\cup$ *Output port facing DAT, such that*:

Input port facing DAT stage = {ctl.t13}

Input port facing DTX stage = {ctl.t1, ctl.t2, ctl.t3}

Output port facing DAT stage = {ctl.t7, ctl.t8}

Output port facing DTX stage = {ctl.t15}

(iii) *Each Input and Output functions for transitions are defined in Figure 4.9 graphically*.

4.3.7 The DAT stage

The DAT stage is a stage that is closest to the resource among all the stages in MDPA. Since DAT also provides data cache, it does not need to access the resources every time. Therefore, we define two distinguished patterns of event processing in the DAT stage: DAT event, and EXT event. For the presentation filtering and event filtering through the data pipeline, these two event processing types traverse stages in a same pattern. However, in DAT stage, DAT event does not access the resources and EXT event accesses the resource. We define the DAT with these different event flows in Figure 4.10.

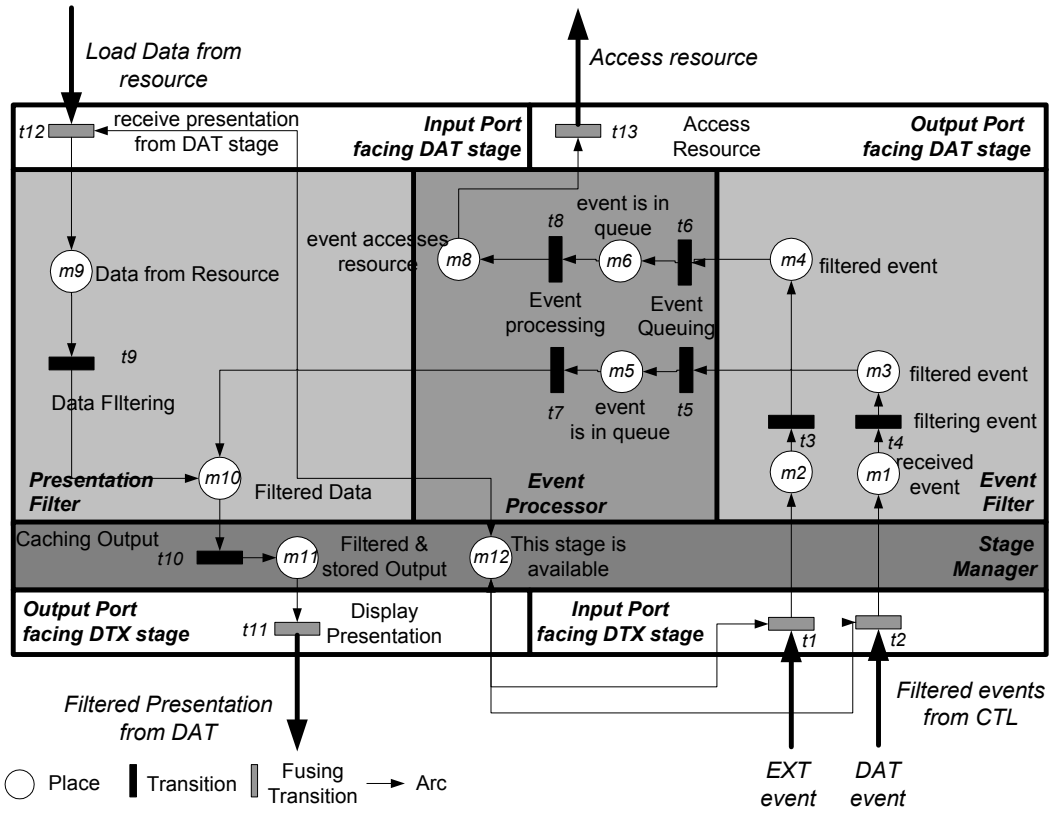


Figure 4.10 PT_{DAT} : DAT stage with PT-net

Stage DAT is defined as,

DEFINITION 4.11 DAT stage *is defined as a PT-net such that:*

(i) $PT_{DAT} = (P_{DAT}, T_{DAT}, W_{DAT}, Mo_{DAT})$,

P_{DAT}, T_{DAT} *are defined in* Figure 4.10 *graphically.*

$\forall\, p \in DAT,\ W_{DAT}(p) = 1$, *constantly.*

$Mo_{DAT}(dat.m12) = 1$, *initially.*

(ii) $Interface_{DAT}$ = *Input port facing RESOURCE*

$\cup$ *Input port facing CTL stage*

$\cup$ *Output port facing RESOURCE*

$\cup$ *Output port facing CTL,* *such that*:

Input port facing RESOURCE = {dat.t12}

Input port facing CTL stage = {dat.t1, dat.t2}

Output port facing RESOURCE = {dat.t13}

Output port facing CTL stage = {dat.t11}

(iii) *Each Input and Output functions for transitions are defined in Figure 4.10 graphically*.

4.3.8 A service in MDPA

Eventually, the service is defined as a Modular PT-net such that,

DEFINITION 4.12 The single service in MDPA is defined as a triple $MN_{Service} = (S, PF, TF)$ *such that:*

(i) *S is a finite set of modules called as "stage".*

$S = \{PT_{DAT}, PT_{CTL}, PT_{DTX}, PT_{PCL}, PT_{MCL}\}$

Where each stage defined as a PT-net such that:

$PT_{DAT} = (P_{DAT}, T_{DAT}, W_{DAT}, Mo_{DAT})$;

$PT_{CTL} = (P_{CTL}, T_{CTL}, W_{CTL}, Mo_{CTL})$;

$PT_{DTX} = (P_{DTX}, T_{DTX}, W_{DTX}, Mo_{DTX})$;

$PT_{PCL} = (P_{PCL}, T_{PCL}, W_{PCL}, Mo_{PCL})$;

$PT_{MCL} = (P_{MCL}, T_{MCL}, W_{MCL}, Mo_{MCL})$;

(ii) *There is not fusing place in* $MN_{Service}$, *Therefore* $PF = \{\}$

(iii) We define *Interface of stage s as a set of fusing transitions in stage s,*

$\text{Interface}_s = InputPort_s \cup OutputPort_s$ *where,*

$InputPort_s$ *is a set of fusing transitions, such that:*

$ip \in s$, $O(ip) \in s'$ *and* $I(ip) \in s'$ *where* $s \neq s'$.

$OutputPort_s$ *is a set of fusing transitions, such that*

$op \in s$, $I(op) \in s$, *and* $O(op) \in s'$ *where* $s \neq s'$

There is a set of fusing transitions in $MN_{Service}$,.

$TF_{service} = Interface_{DAT} \cup Interface_{CTL} \cup Interface_{DTX} \cup interface_{PCL} \cup interface_{MCL}$

The PT-net for individual stages is defined in Definition 4.7, Definition 4.8, Definition 4.9, Definition 4.10, and Definition 4.11.

4.4 Analysis of Design MDPA

We analyze stages with the reachability approach, and simulation. The reachability tree represents the reachability set of a PT-net. The tree represents all possible sequences of transition firings. Every path is a tree, starting at the root and corresponds to a legal transition sequence. A PT-net is bounded if there exists an integer k such that the number of tokens in any place cannot exceed k. This property can be tested by using the reachability tree. If the PT-net is bounded, the PT-net represents a finite state system [Peterson+81].

We attached the results of full script of design and analysis for each stage, in appendix A. Each event types require their unique data flows. Our design of MDPA must satisfy each of these different data flows. Table 4.2 shows the detail descriptions of dataflow required in each event type. This description is organized by the stages and processing steps. The source and destination is the place which the event or presentation starts and arrives for the process. Pre-condition is a set of places which should be marked to fire the process, and Post-condition is a set of places which should be marked after the process.

For two nodes v_s and v_d, we define DPF (v_s, v_d) to denote the set of all directed finite paths connecting v_s and v_d, i.e. all finite sequences of nodes and arcs $v_1,a_1,v_2,a_2,\ldots v_n$ where $v_1 = v_s, v_n = v_d$, and for all i in $1,\ldots n$, $N(a_i) = (v_i, v_{i+1})$ where N is a node function. It is defined from arc into a set of nodes.

Next we consider *Strong Connected Components* (*SCCs)*. Two nodes are *strongly connected*, if and only if, there exists a finite directed path, which starts at a source node and ends in a destination node. From the reachability tree, we can calculate the SCC graph.

Table 4.2 The Description of Detail Dataflow based on Event Types

Event Type	Step	Stage	Pre-cond.	Post-cond.	Source	Destination
MCL event	EF	MCL	mcl.m18	mcl.m18	mcl.m1	mcl.m7
	EP	MCL	mcl.m18	mcl.m18	mcl.m7	mcl.m14
	PF	MCL	mcl.m18	mcl.m18	mcl.m14	mcl.m17
PCL event	EF	MCL	mcl.m18	mcl.m18	mcl.m2	mcl.m8
		PCL	pcl.m17	pcl.m17	pcl.m1	pcl.m10
	EP	PCL	pcl.m17	pcl.m17	pcl.m10	pcl.m12
	PF	PCL	pcl.m17	pcl.m17	pcl.m12	pcl.m16
		MCL	mcl.m18	mcl.m18	mcl.m15	mcl.m17
DTX event	EF	MCL	mcl.m18	mcl.m18	mcl.m3	mcl.m9
		PCL	pcl.m17	pcl.m17	pcl.m2	pcl.m6
		DTX	dtx.m14	dtx.m14	dtx.m1	dtx.m5
	EP	DTX	dtx.m14	dtx.m14	dtx.m5	dtx.m10
	PF	DTX	dtx.m14	dtx.m14	dtx.m10	dtx.m13
		PCL	pcl.m17	pcl.m17	pcl.13	pcl.m16
		MCL	mcl.m18	mcl.m18	mcl.m15	mcl.m17
CTL event	EF	MCL	mcl.m18	mcl.m18	mcl.m5	mcl.m11
		PCL	pcl.m17	pcl.m17	pcl.m4	pcl.m8
		DTX	dtx.m14	dtx.m14	dtx.m2	dtx.m6
		CTL	ctl.m12	ctl.m12	ctl.m1	ctl.m4
	EP	CTL	ctl.m12	ctl.m12	ctl.m4	ctl.m8
	PF	CTL	ctl.m12	ctl.m12	ctl.m8	ctl.m11
		DTX	dtx.m14	dtx.m14	dtx.m11	dtx.m13
		PCL	pcl.m17	pcl.m17	pcl.13	pcl.m16
		MCL	mcl.m18	mcl.m18	mcl.m15	mcl.m17
DAT event	EF	MCL	mcl.m18	mcl.m18	mcl.m4	mcl.m10
		PCL	pcl.m17	pcl.m17	pcl.m3	pcl.m7
		DTX	dtx.m14	dtx.m14	dtx.m4	dtx.m8
		CTL	ctl.m12	ctl.m12	ctl.m3	ctl.m6
		DAT	dat.m12	dat.m12	dat.m1	dat.m3
	EP	DAT	dat.m12	dat.m12	dat.m3	dat.m7
	PF	DAT	dat.m12	dat.m12	dat.m7	dat.m11
		CTL	ctl.m12	ctl.m12	ctl.m9	ctl.m11
		DTX	dtx.m14	dtx.m14	dtx.m11	dtx.m13
		PCL	pcl.m17	pcl.m17	pcl.13	pcl.m16
		MCL	mcl.m18	mcl.m18	mcl.m15	mcl.m17
Ext. event	EF	MLC	mcl.m18	mcl.m18	mcl.m6	mcl.m12
		PCL	pcl.m17	pcl.m17	pcl.m5	pcl.m9
		DTX	dtx.m14	dtx.m14	dtx.m3	dtx.m7
		CTL	ctl.m12	ctl.m12	ctl.m2	ctl.m5
		DAT	dat.m12	dat.m12	dat.m2	dat.m4
	EP	DAT	dat.m12	dat.m12	dat.m4	dat.m8
	PF	DAT	dat.m12	dat.m12	dat.m9	dat.m11
		CTL	ctl.m12	ctl.m12	ctl.m9	ctl.m11
		DXT	dtx.m14	dtx.m14	dtx.m11	dtx.m13
		PCL	pcl.m17	pcl.m17	pcl.13	pcl.m16
		MCL	mcl.m18	mcl.m18	mcl.m15	mcl.m17

First, we analyze each module. Our design should satisfy the constraint that every (source, destination) is SCCs for all processes in Table 4.2. For instance, we examine whether each pair of source and destination has a direct path between them. With analysis tools, Tina Ver. 2.5.1 [LAAS], and Visual Object net++ [Drath], we showed that every (source, destination) is strongly connected. After analyzing the module, we now analyze the linkage of modules. By the definition of interface which is defined as a set of fusing transitions, we illustrate the destination of one stage and source of another stage are strongly connected.

4.5 Summary

In this chapter, we modeled MDPA with the software architectural pattern of Pipes and Filters. Furthermore, every detailed data flow is depicted with the PT-nets model. Finally, the dataflow over the whole pipeline structuring a service is defined as a Modular PT-net.

The modular approach simplifies the design phase and also the analysis phase. We analyzed each stage with the reachability approach. The dataflow through the whole pipeline is analyzed with the modular analysis and definition of fusing transitions between modules.

In the next chapter, we will focus on the scalability and interoperability in MDPA with the development of a collaborative system.

CHAPTER 5 AGGREGATION AND COLLABORATION WITH MDPA BASED SERVICES

A service designed with MDPA contains modular stages which is structuring pipeline for distributed services in P2P Grids environments. Current distributed services are provided as a single service or integrated services. For instance, well known portal services such as Yahoo! [Yahoo], and MSN [MSN], provide multiple services within its integrated user environment. We call this integrating effort as the aggregation of the services.

With MDPA, multiple services are possible to be aggregated in single user's environments. A part of data pipeline is shared by multiple services. In the integrated service, a MCL stage is shared by data pipelines which provide services. Generating user presentation view of MCL stage is shared by multiple services completely. However, the places and transitions related to the event handling should be processed individually, because they have to be defined with its unique functionality of each service. Similarly, the generation of user presentation within PCL also can be shared. We define an aggregated service as a Modular PT-net and show how multiple services are integrated in MDPA.

Next, we describe collaboration models with MDPA in this chapter. The collaborative service is an extreme example of distributed services. While enabling the design of a collaboration service, MDPA should consider the capability of interoperating between services. Furthermore, the ubiquitous environment requires a flexible resource

sharing method, because of the diversity of hardware capabilities and communication environments.

Figure **5.1** shows a snapshot of collaboration in the heterogeneous environments. Heterogeneous devices require factors such as CPU capabilities, network communication, or display types of their specific environments to be taken into account during design decisions. The collaboration in the heterogeneous environments requires diversity in their collaboration session to be managed seamlessly. We introduce collaboration models according to different stages and thus provide various flexibilities in sharing resources.

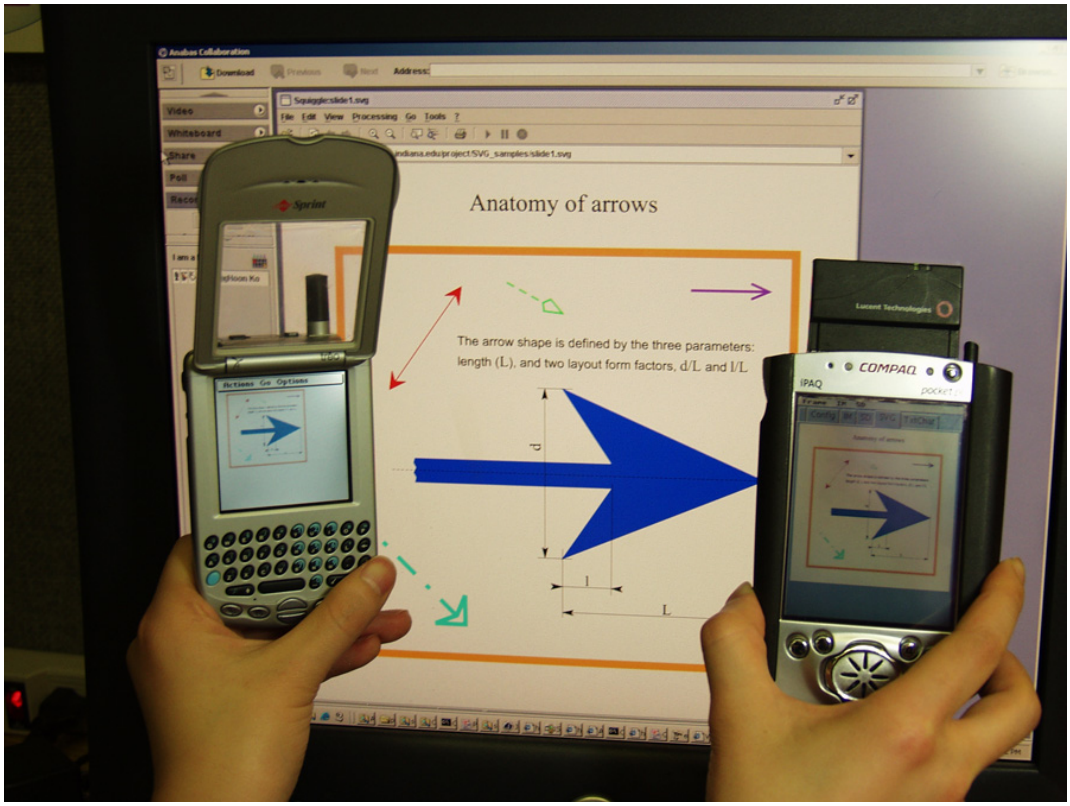


Figure 5.1 Collaboration in Heterogeneous Environments

MDPA is possible to be deployed to current advanced infrastructure, such as Web Service. The Web Service infrastructure provides a universal service description method, such as WSDL [Christensen+01], and data syntax for exchanging information between Web based distributed services [Bray+00]. We illustrate how we deploy the collaborative models with MDPA into Web service infrastructure in this chapter.

This chapter is organized as follows. First, we describe the aggregation of services in MDPA, and illustrate the integration mechanism in section 5.1. Section 5.2 covers the collaboration models in MDPA. We provide descriptions and definitions for each model. Also we discuss various flexibilities in collaboration based on each model.

5.1 Aggregating of Services

MDPA provides scalability with the aggregation of multiple services into a single user presentation. A user is allowed to register with multiple services and browse these services in a single viewer. The aggregation of services is provided by the PCL stage and also by sharing unified MCL stage. Generated presentations in each pipeline of the services are passed to one PCL input as shown in Figure 5.2. Figure 5.2 shows how the user presentation generated for service A, and B are aggregated into one user display.

To aggregate two data pipelines of Service A and B, it requires,

- *Interface facing each data pipeline*: The PCL stage should support sets of interface for each data pipeline. Since input and output depend on the data pipeline, the interfaces must be distinguished. For aggregating of two services named Service A and Service B, PCL stage should support set of input ports facing Service A, and set of input ports facing Service B for presentation generation. There should also be set of output ports facing Service A, and set of output ports facing Service B for the event transmission path.
- *Shared process of presentation generating*: The presentation generating is shared by Service A and B. Therefore, an input from the DTX stage of service A or B, is processed in the PCL stage and passed to the MCL stage.
- *Independent event transmitting to each data pipeline*: However, the event from MCL should be distinguished by its association with specific services. Otherwise, the PCL stage cannot transmit the event through the relevant output ports. Therefore, the user event should not be shared in the PCL stage.
- *Unified MCL stage for a single device*: The integrated service provides a unified user display. Therefore, there is only one MCL stage which browses the display and catches user events.

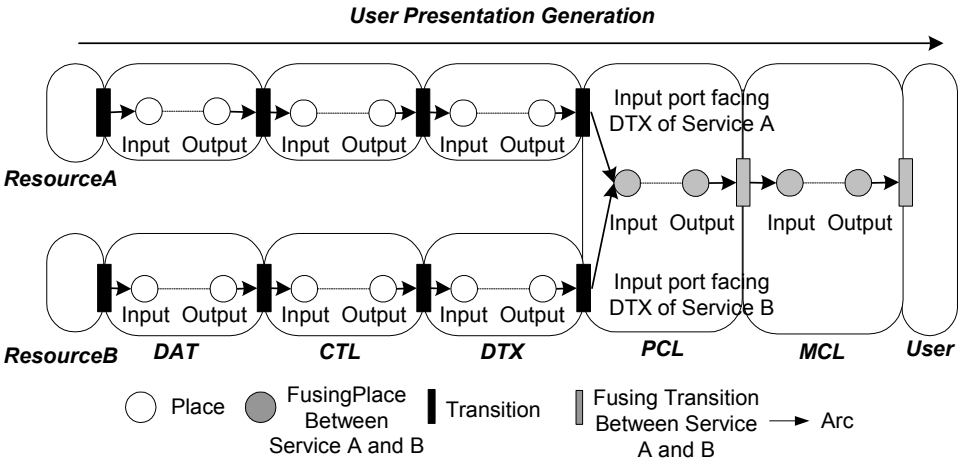


Figure 5.2 Aggregating Dataflow of Generating Presentation

We define an aggregated service as a Modular PT-net based on the definitions of stages in chapter 4.

DEFINITION 5.1 *An* aggregated service *is a modular PT-net is a triple aggregatedMN = (S, PF, TF), satisfying the following requirements:*

(i) *S is a finite set of services which defined as Modular PT-net such that*:

$\forall s \in S$ *, service s is a Modular PT-net defining each single service*:

S = (serviceA, serviceB, serviceC…);

(ii) *PF is a finite set of place fusion sets between services* $\forall s \in S$ *,such that:*

PF = {serviceA.mcl.m10, serviceA.mcl.m14, serviceA.mcl.m15, serviceA.mcl.m17, serviceA.mcl.m18, serviceB.mcl.m10, serviceB.mcl.m14, serviceB.mcl.m15, serviceB.mcl.m17, serviceB.mcl.m18 }

members of a place fusion set have identical initial markings:

$$\forall pf \in PF : \forall p_1, p_2 \in pf : \left[M_0(p_1) = M_0(p_2)\right]$$

(iii)*TF is a finite set of transition fusion sets where*:

$$TF = \{ serviceA.mcl.t10, serviceA.mcl.t15, serviceA.mcl.t16, serviceA.mcl.t17, serviceA.mcl.t18, serviceB.mcl.t10, serviceB.mcl.t15, serviceB.mcl.t16, serviceB.mcl.t17, serviceB.mcl.t18\}$$

Definition 5.1 denotes the aggregating of services in MDPA. The places and transitions related to generating presentation are shared in this architecture. Event transmitting is performed individually by the specification of each service. Note that there is only one PCL and one MCL stage for a user using an aggregated service. Each service shares a set of places and transitions located in the PCL and MCL stages to generate its user presentation view. Meanwhile, event transmitting is not shared. Since each service defines its own event processing which may differ depending on the services, it is not allowed to share the places or transitions related to event transmit. To focus on the generating presentation process, we describe only the user view generation in Figure 5.2.

In a real service, while designing interfaces for different services one should consider the capability of communication and data interoperability. Popular XML technology meets our requirement. Moreover, WSDL like service description languages can provide powerful interoperability to this architecture. We applied MDPA architecture to current the Web Service infrastructure for demonstration purposes.

5.2 Collaboration models and Interoperability in MDPA

MDPA considers the interoperability between users. As we described in previous subsection, an aggregated service for single user is defined with Modular PT-net in Definition 5.1. Now we consider the cooperation between services. Among the distributed services over a P2P Grid, there is a collaborative service for sharing resources among multiple users. Interoperability is an important criterion since the object is shared during traversal over the data pipelines.

There are many styles and approaches to collaboration. In asynchronous collaboration, different members of a community access the same resource. The Web has revolutionized asynchronous collaboration where in its simplest form, one member is posting or updating a web page, and others are accessing it. Asynchronous collaboration has no special time constraints and typically each community member can access the resource in their own fashion; objects are often shared in a coarse grain fashion with a shared URL pointing to a large amount of information. Asynchronous collaboration is quite fault-tolerant as each user can manage their access to the resource and accommodate difficulties such as poor network connectivity. Further, well-established caching techniques can usually be used to improve access performance as the resource is not expected to change rapidly.

Synchronous Collaboration is at higher level no different from the asynchronous case except that the sharing of information is done in real-time. The real-time constraint implies delay of around 10-1000 msec. per participant or rather "jitter in transit delays" of a "few" msecs. Note these timings can be compared to the second or so it takes a browser to load a new page; the several seconds it takes a lecturer to gather thoughts at the start of a new topic; and the 30 msec. frame size natural in audio/video transmission. These numbers are much longer than the parallel computing MPI message latency measured in microsecond(s) and even the 0.5 -3 msec. typical latency of a middlze-tier broker.

Nevertheless synchronous collaboration is much harder than the asynchronous case for several reasons. The current Internet has no reliable quality of service and so it is hard to accommodate problems coming from unreliable networks and clients. If your workstation crashes during an asynchronous access, you just need to reboot and restart your viewing at the point of interruption; unfortunately in the synchronous case, after recovering from an error, one cannot resume where one lost contact because the rest of the collaborators have moved on. Further synchronizing objects among the community must often be done at a fine grain size. For asynchronous education, the teacher can share a complete lecture whereas in a synchronous session we might wish to share a given page in a lecturer with a particular scrolling distance and particular highlighting.

In summary synchronous and asynchronous collaboration both involve object sharing but the former is fault sensitive, has modest real-time constraints and requires

fine grain object state synchronization. We discuss collaboration models for synchronous collaboration with MDPA. Based on the understanding of collaborative service, we also consider the heterogeneity of user devices in same the collaboration session. Different users can require different flexibilities of data accessing.

The motivation of a collaboration model within MDPA is to support a flexible heterogeneous environment for users. An object in MDPA can be shared in different stages. The flexibility of collaboration depends on where the object is shared in the pipeline and how much manipulations data is allowed for the users. By only sharing the display, users cannot expect significant flexibility in sharing resources. Meanwhile, if each user can maintain their own replicate of the service, and exchange only user events, the service can support more flexibility. We present three collaboration models each providing a different depth of flexibility in MDPA: Shared Display Model, Shared Output Model, and Shared Input Model.

5.2.1 Shared Display Model

In the shared display model, participants of Shared Display share the master's framebuffered image and exchange information about the management of states. To support heterogeneous clients, a service requires sophisticated shared display environments to automatically change size and color resolution to suit each community member. For this content adapting process, we can add some filters into Figure 5.3 between the master's output port and the participants' input ports.

Basically, the service is required process which captures the output of master's MCL output object and communication for sending it to the participants. The participant requires a browser, which is able to display the MCL output from master's data pipeline. We define a collaborative service designed based on the shared display model as the Modular PT-net. Each stage of a service in Modular PT-net conforms to the definitions in chapter 4.

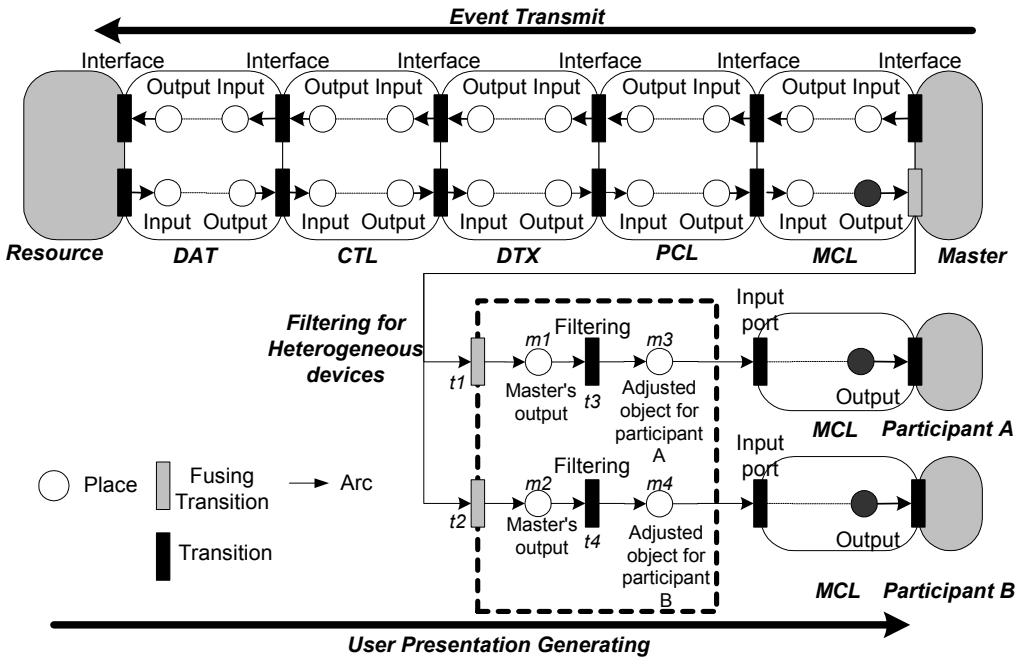


Figure 5.3 Collaboration with Shared Display Model

DEFINITION 5.2. *A collaborative* service using Shared Display Model *is a modular PT-net with tuple of SharedInputCollab = (S, PF, TF), satisfying the following requirements:*

(i) *S is set of Master and Participants defined as Modular PT-net such that*:

(a) Master is a service (DEFINITION 4.12).

(b) Participants is a set of partial services which includes at least MCL stages.

(c) Filters is a PT-net filter = (S, P, W, M), such that,

Set of places and set of transitions are defined in Figure 5.3 *graphically.*

$\forall p \in filter, W_{Filter}(p) = 1$, *constantly.*

$\forall p \in filter, M_{Filter}(p) = 0$, *initially.*

(ii) $PF = \{\}$

(iii) *TF is a finite set of transition fusion sets where*:

$TF = \{Master.mcl.t18, filter.m1, filter.m2\}$

Shared display model has one key advantage – it can immediately be applied to all shared objects. Shared display has two objects; it has two obvious disadvantages - it is rather difficult to customize and requires substantial network bandwidth.

We can suggest two types of solutions,

- *Design MCL stage to perform data compression and customization*. – This method is the most immediate solution for the Shared display's disadvantages. However, developing multi-functional user faced applications is not productive in the current situation where there are a multitude of heterogeneous devices emerging.
- *Design a middleware to support data management* – Practically, the transition between master's pipeline and participants' pipeline is performed by network communication. The method of communication will vary from TCP/IP socket connection to intelligent messaging system. To reduce the workload on the user device and to satisfy the requirements of heterogeneous devices, developing a versatile messaging middleware is important. This middleware is required to support heterogeneous communication environments, data compression for low bandwidth devices, and minimum data customization for a given user's needs.

5.2.2 Shared Output Port Model

Shared output port model only involves a full data pipeline as depicted in Figure 5.4. The output port defined between DTX and PCL stages is shared by participants A and B. Therefore, the participants will get the output of master's DTX stage directly to their PCL stage as if it is their own input data.

One simple example can be built around any content or multimedia server with multicast output stream(s). This method naturally gives, like shared display, an identical view for each user but with the advantage of typically less network bandwidth since the bitmap display usually is more voluminous than the data transmitted to the client to define the display.

With Shared output port model, participants still can expect the content adapting of PCL stage. Although it does not fully support complete data adapting like the stage DTX

stage, participants are allowed to customize their presentation in a limited fashion, such as changing the document format from HTML to WML for mobile devices.

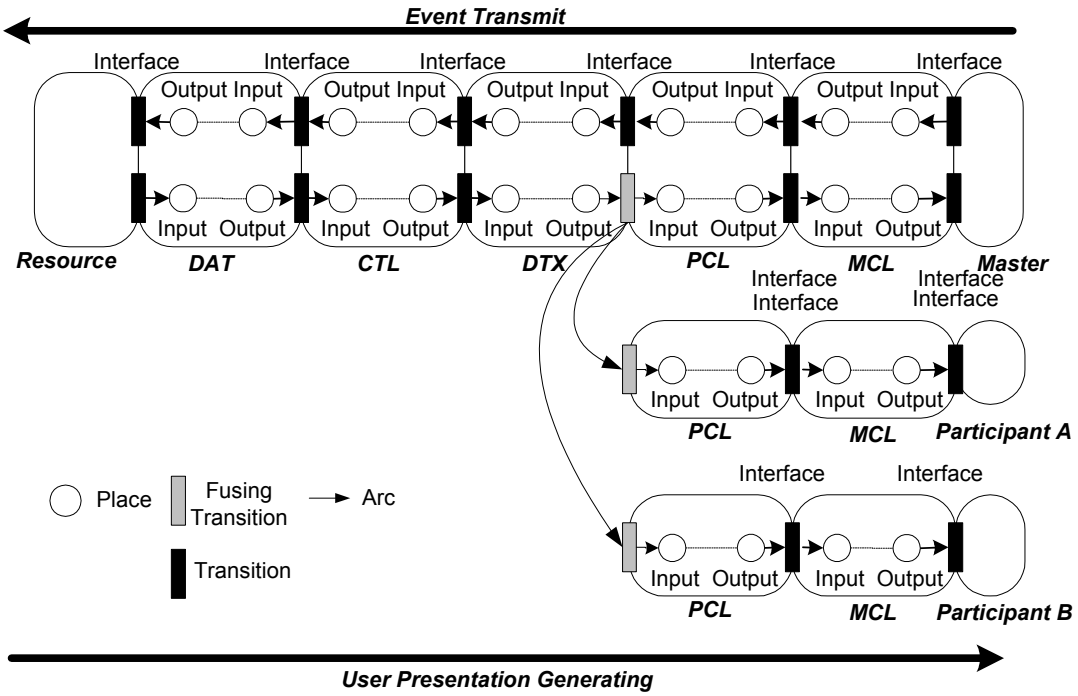


Figure 5.4 Collaboration with Shared Output Model

Based on the design of the service in Chapter 4, we can design the collaborative service using the Output Port model.

DEFINITION 5.3. *A collaborative* service using Shared Output Port Model *is a modular PT-net with tuple of SharedInputCollab = (S, PF, TF), satisfying the following requirements:*

(i) *S is set of Master and Participants defined as Modular PT-net such that*:

(a) Master is a service (DEFINITION 4.12).

(b) Participants are sets of partial services which includes at least MCL and PCL stages.

(ii) *PF = {}*

(iii)*TF is a finite set of transition fusion sets where*:

TF = {Master.dtx.t14, Master.pcl.t18, ParticipantA.pcl.t18, ParticipantB.pcl.t18}

5.2.3 Shared Input Port Model

In the shared input port model, one replicates the pipeline to be shared with one copy for each client. Then sharing is achieved by intercepting the pipeline before the master's pipeline and directing copies of the messages on each input port of the master's pipeline to each of the replicated pipelines. We can illustrate this with a more familiar PowerPoint example. Here all the clients have a copy of the PowerPoint application and the presentation to be shared. On the master client, one uses some sort of COM wrapper to detect PowerPoint change events such as slide and animation changes. These change events are sent to all the participating clients. One can build a similar shared Web browser and for some browsers (such as that for SVG from Apache) one can in fact directly implement the standard of interface defined for the pipeline. However, the input to the DTX stage from the PCL stage should be filtered, since the user presentation is customized for each user in the DTX stage. Therefore, the input can be shared after the filtering in DTX stage of the master's pipeline.

DEFINITION 5.4. *A collaborative* service using Shared Input Port Model *is a modular PT-net with a triple SharedInputCollab = (S, PF, TF), satisfying the following requirements:*

(i) *S is a finite set of services (DEFINITION 4.12) which defined as Modular PT-net such that*:

Each service, $s \in S$, is a Modular PT-net defining each service for Master and Participants:

s = (Master, ParticipantA, ParticipantB,…);

(ii) *PF = {}*

(iii)*TF is a finite set of transition fusion sets where*:

TF = {Master.ctl.t15, Mater.dtx.t15, ParticipantA.ctl.t15, ParticipantB.dtx.t15, ParticipantB.ctl.t15, ParticipantB.dtx.t15,...}

There are advantages and disadvantages in the shared input model. Using the Shared Input Model uses can achieve maximum flexibility in their collaborative services. The replicated service for each user provides a fully functional data adapting process to each user presentation. Also there is the advantage of network bandwidth usage. Since the communication between users are only exchanging shared events, compared to other models, users can perform the collaborative features independent of other participants' network communication environments.

However, there are also disadvantages in the Shared Input Port Model. Compared to the previous models, service providers should provide full data pipelines for each shared application. Therefore this is very time consuming to develop if one must do this separately for each shared application.

Figure 5.5 illustrates the dataflow of sharing of CTL, DAT and EXT event from master. Note that each shared event has to be filtered in the DTX stage of the master's data pipeline.

5.3 Summary

MDPA supports event based distributed services such as a collaborative service. Also MDPA provides the architectural environment of aggregating multiple services into single user presentation view. In this chapter, we presented a formal description of the aggregated service and three major collaboration models. We presented various collaboration models supporting different flexibilities to users: Shared Display model, Shared Input Port model, and Shared Output Port model.

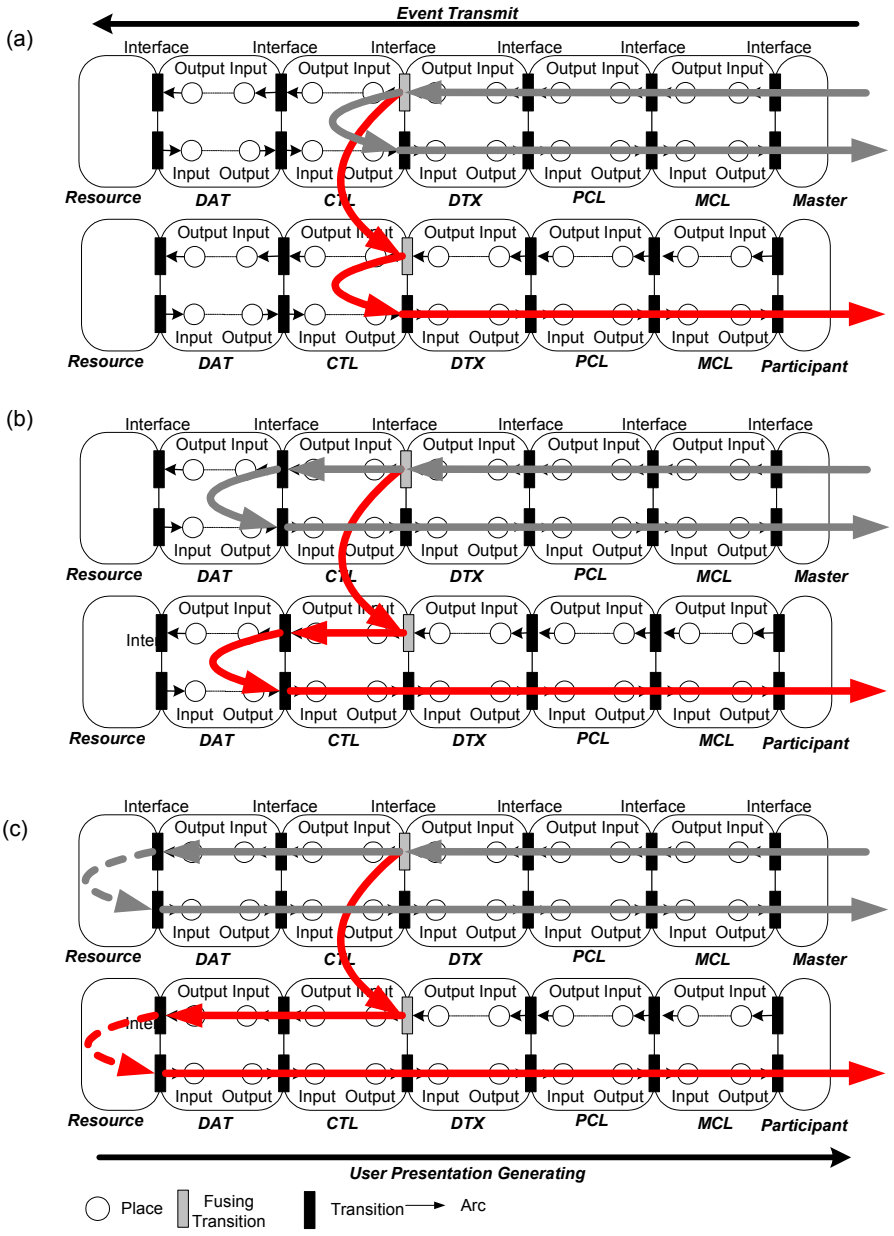


Figure 5.5 Collaboration with Shared Input model

(a) Dataflow in processing collaborative CTL event

(b) Dataflow in processing collaborative DAT event

(c) Dataflow in processing collaborative EXT event

CHAPTER 6 ADAPTATION STRATEGIES TO INCORPORATING CONTENT ENABLE EFFICIENT INTERACTION

The availability of miniaturized devices and wireless communications has made mobile computing feasible. MDPA is motivated by the requirement of adapting heterogeneous devices as client devices into the P2P Grid service. An ever more mobile workforce, and the computerization of inherently mobile activities is driving a need for applications to be integrated with traditional distributed systems. Mobile cellular telephones are widely available these days. A handheld computer integrated with such a telephone is called as a *smartphone*. Currently wireless network service vendors have introduced a wide bandwidth telephone network, 3G communication [Sprint], and it enhances the possibility of adapting a smartphone as a client in traditional distributed systems. Existing wireless LAN technology is already equipped to PDAs and laptops, and enables *pervasive computing* with its wide-bandwidth network communication [IEEE+95]. Pervasive or ubiquitous computing is a new emerging computing style, which adapts various computing devices throughout our living and working spaces. These devices include PDAs, smartphones, traditional desktops, wearable computers and so on. These devices coordinate with each other and network services seamlessly.

In this chapter, we describe our approach to adapt heterogeneous devices into collaboration services while presenting a prototype of the CAROUSEL Web Service. We cover issues that range from transcoding methods to design strategies while combining various adapting technologies for collaboration services. The rest of this chapter is

organized as follows: Section 6.1 describes the steps of adapting heterogeneous devices as a client device in the Carousel Web service. We explain content adapting strategies based on the characteristics of collaborative features in section 6.2. The summary of this chapter is followed in section 6.3.

6.1 Adapting Content

The major process of content adapting is performed in the DTX stage. Sometimes, multiple processing is required for adapting data. For example, resolution adjustment, color adjustment, and data compression can be required for a single data object. However, there are also expensive or rarely used content adapting technologies in the multimedia area, such as special effects or converting text to voice, etc. This can be quite a complex problem by needing to maintain these complicated processes for every service instance. We suggest that the service should access these remote technologies as a service requester.

Figure 6.1 shows how multiple services can access remote technologies. We can consider remote technologies as shared resources. Therefore, again, in the design of the service with Modular PT-net, the shared resources can be denoted as fusing places [Christensen+00]. The concurrency problem is considered in accessing remote technologies A, B, and C for the service I and II.

The design of the content adapting process requires proper selection of technologies based on the characteristics of the application. We investigate the utilization of various transcoding technologies in this design of collaboration services. In the shared display model, one shares the bitmap display and the state is maintained between the clients by transmitting the changes in the display. Meanwhile, the shared input/output port model filters the output of each application to one of a set of common formats and builds a custom shared event viewer for these formats. This allows a single collaborative viewer to be leveraged among several different applications. Document formats such as W3C's Scalable Vector Graphics (SVG) [Ferraiolo+01] or Adobe, Inc.'s Portable Document Format (PDF) [Adobe] are particularly interesting and the support of collaborative

viewers is a major advantage of the CAROUSEL Web service. The scalability of vector graphics, and separation of user presentation from the master's content, enables the shared input/output port model to provide more flexibility for scientific visualization or geographical information systems. The constraints of "real-time synchronous collaboration" for both methods, implies a delay of 10-100 millisecond for each participant [Fox+02a]. In shared display method, the time constraint is the most important factor to perform its illusion of collaboration. Basically shared input/output port model also requires this "real-time" constraint; however the fidelity of shared resource has a higher priority within the range of time delay for synchronous collaboration. We investigate the technology for adapting content that can maximize the collaborative features and optimize network bandwidth. In section 6.4, we discuss the content adapting based on the collaborative features.

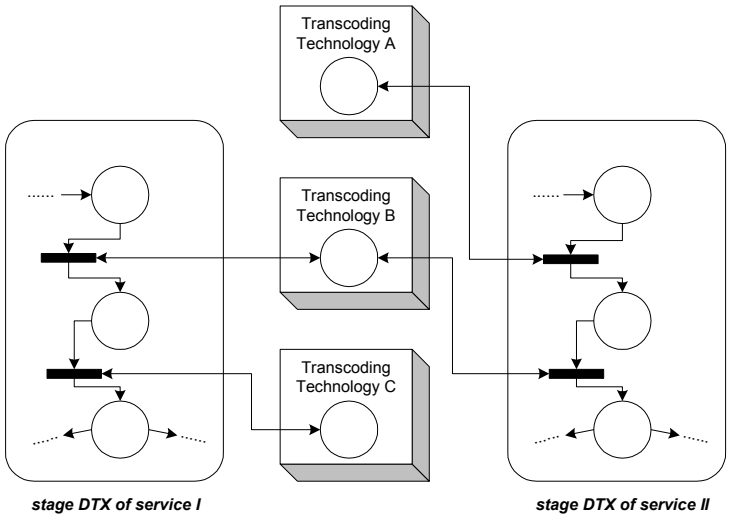


Figure 6.1 Sharing remote content adapting technologies

6.2 Application -aware Transcoding

Originally, transcoding technology has been considered for converting multimedia from one format to another format preferred by specific devices [Chandra+00]. We utilize various transcoding technologies to adapt shared content based on the type of collaboration. Here

transcoding technology includes image resizing, converting image formats, compressing data, and transform technology such as the use of stylesheets.

Transcoding is one of the most popular ways to tune content from a service provider. Transcoding is the transformation that converts a multimedia object from one form to another, frequently trading object fidelity for size, and is used to convert image or video formats (reducing resolution or compressing data) [Chandra+01]. A classical approach from ORL AT&T Lab's VNC [AT&T] allows user to customize encoding methods. As extended versions of VNC for mobile devices, there are Harakan Software's PalmVNC [Harakan], Nokia's Java VNC viewer [Nokia]. These approaches are implemented on the client side of the VNC protocol for PDAs and wireless phone. They enabled a mobile device to be a thin client of a traditional desktop coupled with the sharing of master's view and the execution of user events.

Transcoding technology is also used to fit document and graphics files to the unique constraints of mobile devices and other Web-enabled products. A number of distributed services use transcoding technology to generate documents for their heterogeneous clients [Chandra+01, Intel, Oracle, IBM+02b, Smith+99]. The Apache Cocoon [Apache+99] project allows automatic generation of HTML, PDF, and WML files through the processing of statically or dynamically generated XML files using XSL [Adler+01] and XSLT [Kay+03]. The idea of transcoding is also adapted to Web service architectures. Duke University's Quality Aware Transcoding project [Chandra+00] investigates differentiated Web services, which enables the Web services and Web servers to manage their available bandwidth with its quality aware transcoding. IBM introduces *WebSphere Everyplace Access* [IBM+02b], which supports developers and administrators to utilize the transcoding technology by accessing portlet which performs transcoding. Transcoding technology is developed as a component of WebSphere, which is IBM's Web service infrastructure [WebSphere]. *WebSphere Everyplace Access* is designed to perform its transcoding process in its individual portlet. This approach is different from existing distributed systems supporting the transcoding process for heterogeneous devices, because it enables the transcoding process to be separated from the proxy architecture. It also enables content providers to provide high-quality transcoding technologies on the server side.

For collaboration services, different collaborative features may need different transcoding technologies according to their unique functionality. In this Chapter, we will explain our approach based on two different resource sharing features, shared display and shared export. As an implementation of Shared Display Model, shared display is the simplest method for sharing

documents with the frame-buffer, corresponding to either a window or the entire desktop, replicated among the collaborating clients. Shared display does not allow significant flexibility; for instance, different clients cannot examine separate viewpoints of a scientific visualization. More flexible sharing is possible by sharing object state updates among clients with each client being able to choose how to realize the update on its version of the object, a process known as the shared export mechanism which may be an implementation of the Shared Input or Output Model [Fox+02a].

Shared display requires fast and efficient propagation of the master's change of display. Thus reducing the size of data over the wireless network is critical. On the other hand, shared export provides more flexible visualization and needs to ensure quality of the graphic and user interactivity to provide a user's preferred view.

6.2.1 Measuring Fidelity of Shared Information

For the sharing of resources in heterogeneous environments, the fidelity of the original resource and the efficient transmission of data are tradeoffs in the real-time collaboration service. Fidelity measurements have been done in data compression for image or vedio[Mohan+99]. Distortion of compressed data is typically measured as the mean squared error (MSE) between the source and its compressed version. One problem with the MSE based distortion measure is that it may not correspond to the perceived loss of fidelity [Jayant+93]. However, a bigger drawback is the difficulty of formulating a meaningful distortion measure when the adaptation is drastic. For example, it is difficult to measure the loss of fidelity when document based graphical information is transcoded with stylesheet, or transcoded to its textual alternatives. To resolve this problem, Mohan defines the fidelity of transcoded information in [Mohan+99] and provides approache to measure the fidelity of content adapting.

$V(M_{ij})$ = (perceived value of transcoded version Mi) / (perceived value of original M_{i0})
$V \in [0,1]$, $V = 1$, for original item M_{i0} , $V = 0$, when the item is excluded M_{imi}

Where, $M_i = \{M_{ij}\}$, $j = 0, m_i$, is computed by transcoding A_i into j versions with different resolutions and modalities. The original version is denoted as M_{i0}.

The benefit of V is that we have a measure for fidelity that is applicable to transcoding of media at multiple resolutions and multiple modalities. This also allows us to compare document items that were in different media types. Measuring fidelity is useful to analyze various dynamic content adaptation policies.

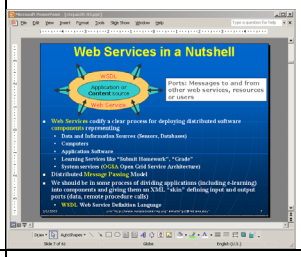
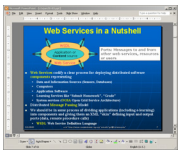
6.2.2 Transcoding in Shared Display

Since Shared Display is sharing documents with the frame-buffer, modest client dependence is possible with mobile devices, for example receiving a reduced size image. Some collaboration systems support remote manipulation with user interactions on one machine holding a replica frame-buffer transmitted to the instance holding the original object. This is an important capability in help desk or consulting applications, similar to situations that occur frequently in the debugging of code.

The shared display processes the image captured from the master's framebuffer in three ways. First, it resizes the images based on the device profile. Table 1 shows a resultant data of a test session from the Garnet collaboration service with Microsoft PowerPoint. Garnet is our earlier prototype of the collaborative system [Fox+02b]. The resolution is customizable on the client side. Second, shared display should support graphic format transformation for specific environments, for example Mobile Information Device Profile (MIDP) from Sun supports only raw bitmap data and the PNG (Portable Network Graphics) [Adler+96] graphic format. To deal with this case, bitmap based Java canvas image object, or GIF, JPEG image should be transformed to PNG image for MIDP equipped smartphone clients. Finally, the compression of image data should be supported to save network bandwidth. We demonstrate the compression of image data with Huffman [Huffman+52] and LZ77 [Storer+82] algorithm in table 6.1. For raw bitmap data, comprising of 8 bit of RGB and alpha, the demonstration in Table 6.1 eliminates alpha bits which represents transparency of image, and reduces the communicating data size to 25%. Table 1 presents the actual data size transformed via a wireless LAN network or 3G communication service as well.

As shown in Table 6.1, the shared information for iPaq users and Treo users provide very low fidelity of the original information. However, with the decrease of resolution and the compression of data, the amount of transmitted data decreased to 1.2 ~ 0.004 % of original image. Although there exists the trade-off between the fidelity and transmitted data size, the user can choose their preferences based on condition specified within the constraints of their device capabilities.

Table 6.1 The resultant data from test session of Garnet shared display

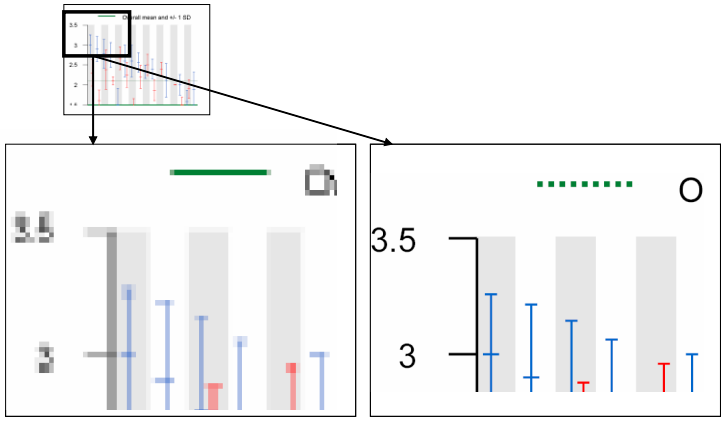
Hardware Specification	Desktop PC 1280 x 1024 LAN SUN J2SE	PDAs (iPaq 3900) 240 x 320 Wireless LAN 802.1b SUN PersonalJAVA	Smartphones (Treo300) 160 x 160 3GWirelessCommunication SUN MIDP
Image			
Image Size (pixels)	1106 x 930	553 x 465	120 x 100
Transmitted data size (KBytes)	4017.9 KB	50.9KB (1.2 % of original image)	1.7 KB (0.004 % of original image)
Image Format	Bitmap image	Bitmap image	PNG image
Fidelity Value	1.0	0.125	0.0043

6.2.3 Transcoding in Shared Input Port model

Since the shared input port model allows users to access more abstract stages of an object, it provides significant flexibility with view points based on user preferences [Lee+02a]. For instance, different users can browse a display data that is generated from identical data sources but is rendered in different ways. This is very time consuming to develop if one must do this separately for each shared application. The shared export model filters the output of each application to one of a set of common formats and builds

a custom shared event viewer for these formats. This allows a single collaborative viewer to be leveraged among several different applications.

Scalability implies that each client can resize and scroll while preserving correct positions of pointers and annotations for their various resolutions. As depicted in Figure 6.2, the benefit of the vector graphics format is its scaling/resizing ability. For mobile devices which have limited display size, this ensures flexible resource access based on user preferences. SVG is useful as it is already available for Adobe Illustrator and both PowerPoint and Macromedia Flash are exportable to this syntax. Currently there is a Flash (which is a binary 2D vector graphics format) to SVG converter [Pronet+00] from the University of Nottingham while OpenOffice.org's *OpenOffice* [OpenOffice] exports PowerPoint to SVG.



(a) (b)

Figure 6.2 400 % Zoom in Images in Bitmap and Vector graphics

Another advantage of SVG is its formatting effects with stylesheets. SVG inherits XML's styling technology such as the use of CSS syntax and properties or XSL. Figure 6.3 shows the different output images rendered with different stylesheets. The advantage of styling with stylesheet is flexibility in reformatting images. In addition to color adjustment, Figure 6.3 - (b) is transformed by CSS stylesheet, which redefines the width of a brush stroke to be wider and the text style to more recognizable.

To provide individual presentation to each user, Garnet allows every user to own their object instance in shared export [Lee+03b]. Figure 6.4 is the object flow of the SVG shared export for each user. Image rendering and transformation are performed in the content server of the CAROUSEL Web service to reduce the workload in mobile devices. Eventually the ready-to-use image is delivered to PDAs or smartphones. There are several graphic formats supported in mobile devices. SUN's MIDP supports only PNG and raw bitmap graphics, while SUN's Personal Java runtime environment supports JPEG and raw bitmap graphics. Thus, we provide format conversion for specific mobile environments. Some of the graphic formats include data compression mechanisms; however, raw bitmap image data needs additional data compression to better utilize wireless network bandwidth. Data compression capabilities can be fine tuned by individual users based on their needs.

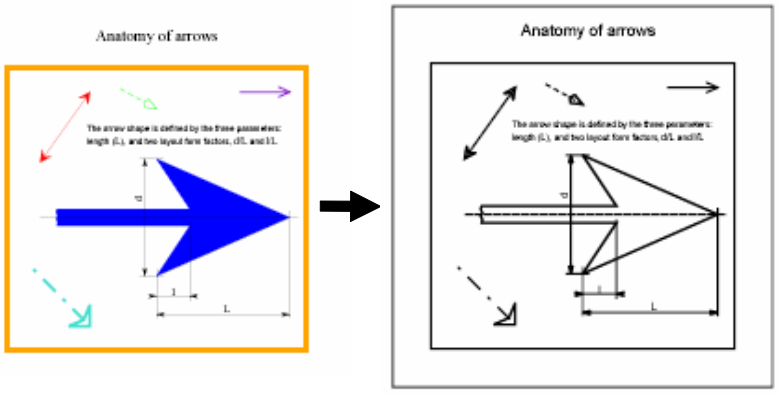


Figure 6.3 Styling with CSS for Black and White PDAs

Since the Shared Input port model is designed for more flexible and high quality resource sharing, stringent timing constraints are not the overriding factors. Along with the scalability of vector graphics, shared export provides maximized resource sharing to users. The users are able to browse the best quality of image supported by their devices.

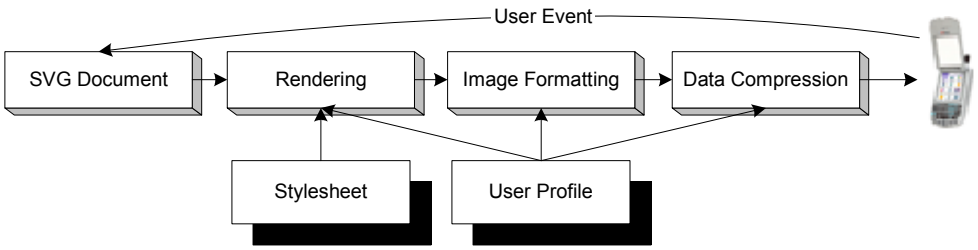


Figure 6.4 Object flow of collaborative SVG

6.3 Summary

Adapting new devices are considered as a stage of major data processing in MDPA. In this chapter, we described the methods of adapting new devices provided in MDPA. Adapting new devices is dealt with in the DTX, and the PCL stage. The process in DTX stage contains major content adapting such as transcoding, and transformation, etc. Meanwhile, the process in PCL is higher level of filtering such as transformation of markup languages.

In the collaboration service, shared display model is used for features, which require immediate response to users. Therefore, the faster response from service is more important even though the shared information provides lower fidelity of the original information. Meanwhile, the shared input port model provides better flexibility to share resources. Therefore, higher fidelity is more important than response time within the constraints of synchronous collaboration. We described how we support user's requirements in different application features.

CHAPTER 7 DEPOYING MDPA AS WEB SERVICES: CAROUSEL WEB SERVICE

As we described in section 5.2.1, in the shared display model with MDPA, participants shares the bitmap data and the state of the master user by transmitting (with suitable compression) the changes in the display of the master's framebuffer. As with video compression like Moving Picture Experts Group (MPEG), one uses multiple event types with some defining full display and others just giving updates. Obviously, the complete display requires substantial network bandwidth but it is useful every now and then, so that one can support clients joining throughout a session, has more fault tolerance and can define full display update points (major events) where asynchronous clients can join a recording. Supporting heterogeneous clients requires that sophisticated shared display environments automatically change size and color resolution to suit each community member. Figure 7.1-(a) illustrates that the Shared display model with a Web service requires an intelligent and powerful event-based messaging system. The event system should provide the data compressions based on the MIME type of data for collaboration service. Moreover, the data compression should be based on the user's profile or preferences. Shared display has one key advantage – it can immediately be applied to all shared object. For existing Web service, we can apply the shared display method for providing collaboration service immediately.

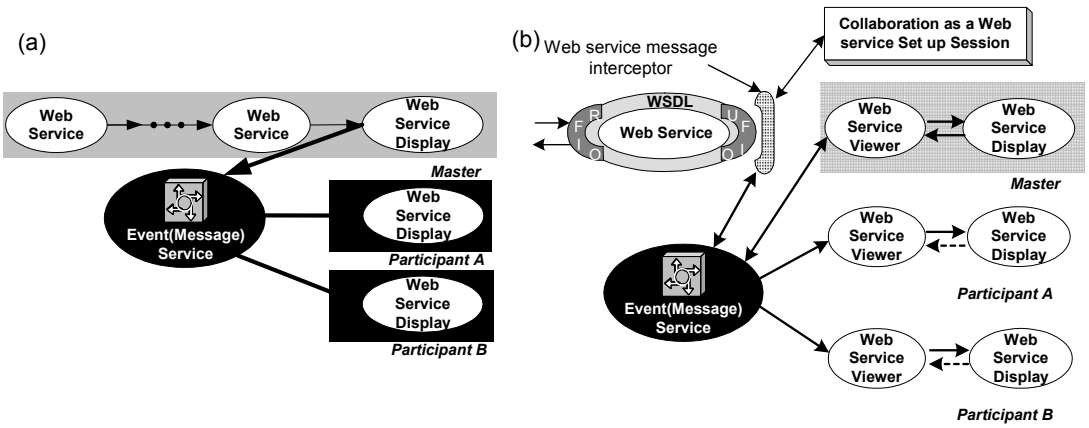


Figure 7.1 Deploying Collaboration model as Web Service

(a) Shared Display model as Web Service

(b) Shared Output model as Web Service

As we presented in section 5.2.2, only a single Web service is involved in the Shard Output Port model. Since the user-facing ports of a Web service define the user interface, this mechanism simply gives a collaborative version of any Web service. In this model, we also require the event service and the message interceptor as depicted in Figure 7.1 – (b). The output data from a Web service is intercepted by the message interceptor and multicast to each participant of the session. Like the Shared display model, the Shared Output Port Model also shares identical user presentation views between participants. However, the Shared Output Port Model supports more flexibility. Since the Web service message interceptor catches the data between user facing output ports defined by the Web service and the aggregator, the user can inherit the benefit of Web service infrastructure such as flexible generation of user presentation view such as translating markup languages from HTML to WML for some mobile devices, or changing presentation layouts. Also, the developer can provide a setup environment for setting up the collaboration session as a Web service portal [Lee+03a].

Meanwhile, the Shared Input Port Model requires individual replications of the Web service for each client. The event service delivers the collaborative event input from the Master's Web service to the participants' Web service. Each Web service generates a user presentation view based on the arrived of a collaborative input event. In the Shared Input Port Model, only the user facing input ports are shared. However, in MDPA architecture, the input port sharing should consider the adjustment of the object in the

DTX stage. The data object is tuned for each heterogeneous device in the DTX stage. Therefore, the user event should be filtered based on the processing of the DTX stage. The collaborative input event also should be filtered to be processed correctly in different user environments for participants.

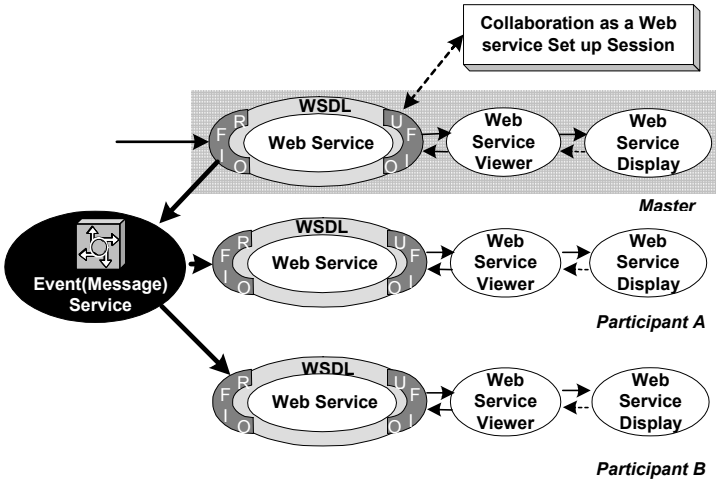


Figure 7.2. Shared Input model as Web service

As a proof of concept, we developed a collaboration service as a Web service with a collaborative model: CAROUSEL Web service. We present the architecture and components in next subsection.

7.1 The CAROUSEL Web Service

As we presented in chapter 3, the Web service infrastructure provides a standardized environment to deploy MDPA. Using XML [Bray+00], WSDL [Christensen+01], or a SOAP [Box+00] like standard provides the interoperability between services. Grids like shared resources are integrated and provide the distributed service which is sharing the largely asynchronous resources. The current Web service infrastructures such as the Apache project's Jetspeed [Jetspeed] or WebSpheres [Webspheres] from IBM provide services and development tools for mobile devices. However, these approaches based on

portal services are not efficient to support real time services such as synchronous collaborative services.

We present the CAROUSEL Web service as a prototype of collaboration based on MDPA. For demonstrative purposes, we have developed a collaborative SVG browser as a feature of the Carousel Web service. We present features of the collaborative SVG browser in this section. SVG is a 2D vector graphics standard format from W3C and has a structured XML syntax [Bray+00]. We have adapted SVG as a format for shared export within our collaboration research. The user-interface will display an SVG image rendered by the content server, provide a display customization environment and process basic collaborative functionalities.

It is approached based on the collaboration models we presented in section 5.2. Especially, we present a prototype of input port collaboration, shared SVG. Clients on heterogeneous devices each get a different presentation view of the replica of each Web service. Moreover, each user will get non-collaborative services, such as individual scrolling or zooming image, customized based on their user profiles. The object sharing in our model of Figure 5.5 is achieved by intercepting the pipeline before the "master" Web service and directing copies of the messages on each input port of the master Web service to the replicated copies. To support devices with limited capabilities, such as PDAs or smart phones, the most elaborate processing is implemented in the content server, and only ready-to-use data is delivered to users. For instance, each user has their object instance for the service with non-collaborative input/output ports, and every presentation view is customized based on the specified user profile. However, the client device only keeps thin client which can perform the MCL stage. The customizability is provided in this example as each client has the freedom to accept or reject data defining the shared Web service.

To satisfy the requirement of sophisticated collaboration, the collaborative system should consider sharing resources in various ways. The resources here include many kinds of objects such as databases, data streams or visualizations. Our approach originates from the refinement of this data processing. Each object is processed in a well-defined dataflow, MDPA, which is designed to facilitate flexible data management. Each stage of the pipeline is a Web service with data flowing from one stage to another.

The Carousel Web service contains four major components: content servers, aggregator, client application and event service. To provide rich synchronous/asynchronous collaboration features we build a message based collaboration architecture. The set of cooperating services in our framework are deployed using the message service for communication. Each component is linked together with their input/output ports designed to support Web services semantics.

7.1.1 Content servers

Each collaborative feature of the CAROUSEL Web service is designed as individual remote content servers in

Figure **7.3**. The major requirements of content servers are,

- Content generation/processing for collaborative/non-collaborative features
- Ports facing towards resources
- Customizing content processing for pervasive users

The requests from the user to the content server can be classified as collaborative and non-collaborative requests. The bitmap based user output can easily be managed on a given user interface, while the output rendered from the filtered document should process every user command in rendering units for its quality of service. For instance, the SVG content server generates and delivers a new SVG image when a user requests a zoomed image only for his or her interest but not for sharing. Otherwise, every user application must keep its own copy of original documents and the processing unit too. Every collaborative and non-collaborative event is transmitted via the NaradaBrokering messaging service.

The content server is designed as a portlet comprising Web services, and provides input and output ports. Each portlet in the CAROUSEL Web service defines distributed objects in a XML-based IDL (Interface Definition Language) called WSDL [Christensen+01]. The overall structure of every message, including input and output, are defined in WSDL. The output/input ports support dynamic communication channels to NaradaBrokering and general HTTP communication. In order to customize the portlet

presentation for each user on pervasive devices, the content server allows any number of transformations. The stylesheets specially designed for universal devices map the original document to the customized one. Every customization is performed based on the user profile from the client.

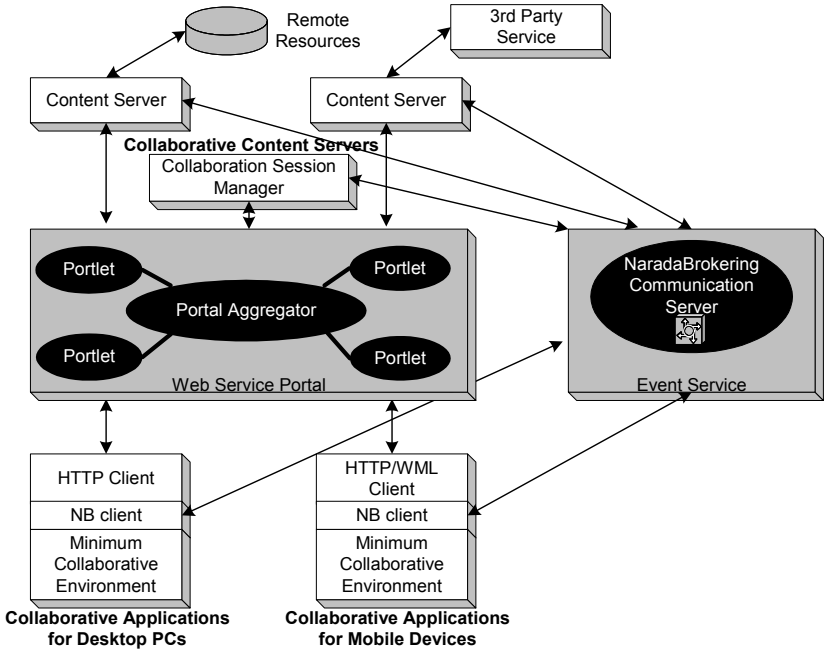


Figure 7.3 Architecture of the Carousel Web service

7.1.2 Event service

The CAROUSEL Web service is a message-based collaborative system. To provide messaging between heterogeneous user network environments and Web services, NaradaBrokering [Fox+02b] from the Community Grid Labs is adapted as a general event brokering system. NaradaBrokering supports centralized, distributed and peer-to-peer (P2P) messaging models with a dynamic collection of brokers supporting a generalized publish-subscribe mechanism. NaradaBrokering can operate either in a client-server mode like JMS [SUNb] or in a completely distributed JXTA-like peer-to-peer mode [SUN+03]. By combining these two disparate models, NaradaBrokering can

allow optimized performance-functionality trade-offs for different scenarios. At the transport level NaradaBrokering provides support for TCP, UDP, Multicast, SSL and RTP. NaradaBrokering's cluster-based architecture allows us to support client configurations that can scale to arbitrary size. For the remote resources behind of the firewall, NaradaBrokering provides the capability to communicate across firewall/proxy boundaries. We expect that the collaborative system developed based on NaradaBrokering's messaging infrastructure will provide the collaboration service, to the users in heterogeneous network environments, with more reliability within a scalable network framework.

For mobile users, such as PDAs and smart phone, HHMP (HandHeld Message Protocol) [Fox+02c] will be integrated to NaradaBrokering as a service in its transport layer. Mobile clients have modest performance and size in comparison with traditional desktop machines. Therefore, they require particularly efficient protocols. HHMP is an efficient lightweight protocol. Furthermore, HHMS's communication protocols are designed keeping in mind specific mobile devices. For example, a wireless Internet accessing service for smart phone or PDA phones supports only HTTP, which entails request-response based transmission primitives currently. HHMS will provide virtual two-way transmission primitives for collaborative network communication environment to these limited devices. Each communication service is selected from the user profile of the client's specification, and assigned to the client automatically.

7.1.3 Client application

The client application is designed to entail minimal data processing. Therefore, the MCL stage is processed as a client application. The customized output data is delivered and displayed through the client application. The major features of the client application are -- user specification, display portlet presentation, and processing user input events.

In the first step, pertaining to the user specification, every user can setup their working environment for their specific machines as well as preferences in the portal presentation view. The operating systems, display types, communication methods and preferred resolutions are the basic factors selected in this phase.

After the initial setup, the collaboration features customized with the user profiles specified in client's setup are provided. Collaborative events such as a major presenter's zooming in on a SVG image or changing new URLs are wrapped with the collaborative application protocol and the event message is delivered to content servers via event services as an XML message. The non-collaborative events are delivered to content servers with the information which identifies the type of the event.

7.1.4 Aggregator

Several collaborative features designed as content servers and supporting services, are implemented as portlets and can be aggregated within a portal. The user sees the portal presentation view, which is the aggregation of the presentations of each portlet. Therefore, aggregator of Carousel Web Service is based on the processing of the PCL stage. The portlet presentation views can be in separate windows and can either be distinct or partially layered on top of each other. This portlet presentation view includes all kinds of machine-dependent outputs such as bitmap display or audio streams.

7.2 Universal Access in Carousel Web service

Universal access in the Carousel Web service is approached by having the user output/input defined intelligently by an interaction between the user "profile" (specifying user/ client capabilities and preferences) and the semantics of general Web services [Booth+03]. The service itself specifies the most important parts of its user-facing view and the output should be modified for clients with limited capabilities. This implies that the data processing in services is deficient in the sense that there must be a clear flow not only from the "basic Web services" to the user, but also back again. This can be quite complicated and it is not clear how this is achieved in general as the pipeline from Web services to user can include transformations, which are not reversible. For this reversibility problem, the content servers of Carousel Web service are designed such that it keeps the original document in itself, and provides an interface to the original document to generate a new output for each event. The ambiguity of reverse

functionalities still exists, but we can expect that every reverse function will get correct output with this design.

In WSDL, the inputs and outputs of operations are termed as ports. The Carousel Web service is designed with one or more ports in each Web service component to provide a general approach as a collaborative application. Here we will discuss how we linked each Web service component via these input and output ports and approach the universal access in this object flow of Web services.

Each Web service is designed with three major user-facing ports as an output port of the modular pipeline of Figure 7.4. First, there is the main user-facing specification output ports that in general do not deliver the information defining the display, but rather a menu that defines many possible views. A selector in Figure 7.4 combines a user profile from the client (specified on a special profile port) with this menu to produce the "specification of actual user output" that is used by a portal, which aggregates many user interface components (from different Web services) into a single view. The result of the transformer may just be a handle, which points to a user facing customized output port. This output port allows users to select user interface components - operating systems, display types, and resolution preferences.

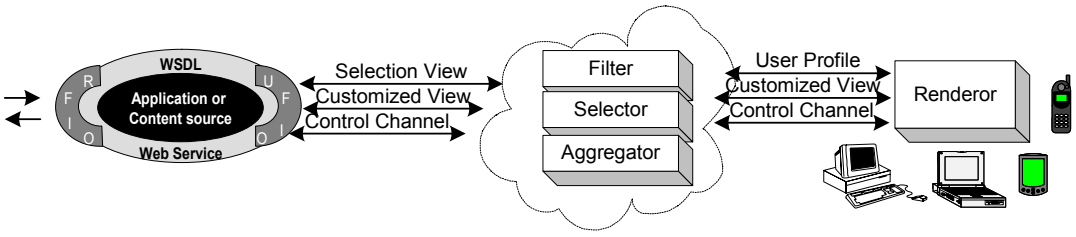


Figure 7.4 Dataflow in Carousel Web service to support Universal Access

Second, there is the customized user-facing output port that delivers the selected view from selector of the Web service to the client. This in general need not pass through the portal, as this only needs the specification of the interface and not the data defining the interface. For collaborative SVG, specification of the output port could involve a choice of display type or resolutions. Rendering an image from an SVG document

transformed with CSS stylesheets [Lie+96] is a customized user–facing output presentation. The conversion between stylesheets could in fact involve a general filter capability of the event service as another Web filter service. It seems appropriate to consider interactions with user profiles and filters as outside the original Web service, since they can be defined as interacting with the service using a general logic valid for many originating Web services.

Finally, we have user-facing input/output port, which is the control channel, shown in Figure 7.4.

Note that in the Carousel Web service, we have lump a portal (such as Jetspeed [Jetspeed] from Apache) as part of the "event service" as it provides a general service (aggregating user interface components) for all applications (Web services). This packaging may not be the most convenient, but architecturally, portals share features with workflow, filters and collaboration. These are services that operate on message streams produced by Web services. Considering universal access in this fashion could make it easier to provide better customizable interfaces and help those for whom the current display is unsuitable.

7.3 Content Adapting in CAROUSEL Web service

MDPA is designed to adapt shared content for heterogeneous devices. Moreover, the process related adapting content is based in the DTX stage and the PCL stage. The content adapting process is done based on the predefined device profile, user's preferences, and the characteristics of service features.

Predefined device profile: These may be several client devices utilizing distributed services. This may range from the wall screen to limited smartphone devices. The device profile keeps information about the individual user device such as display types, operating systems, CPU capability, etc.

User preferences: Services should consider the user's preferences. For example, some users may prefer the more detail information even if it would cause longer transmission time during responses. On the other hand, some users may prefer more

abstract information, as opposed to detailed descriptions, for efficient information exchanges.

Characteristics of application: As we mentioned earlier in this chapter, we should consider the purpose of the service feature. Each feature has a different purpose, and the service should provide appropriate content adapting technology for each feature.

7.3.1 Workflow of CAROUSEL Web Service

A predefined device profile is stored in the distributed database system. In the setup session, users can change their device profile, and the device profile is updated by the information from the setup session. The device profile contains, display type (resolution, color depth, etc.), proxy utilization, network protocol, network bandwidth, operating system, hardware capabilities (memory, CPU capability, etc.), version of script or markup language, and service vendor, etc. Similar to other components, the information described in the profile must be ready for processing by any machine or application. The device profile is represented with XML. The information in the profile can be accessed for processing in the DTX stage and the PCL stage.

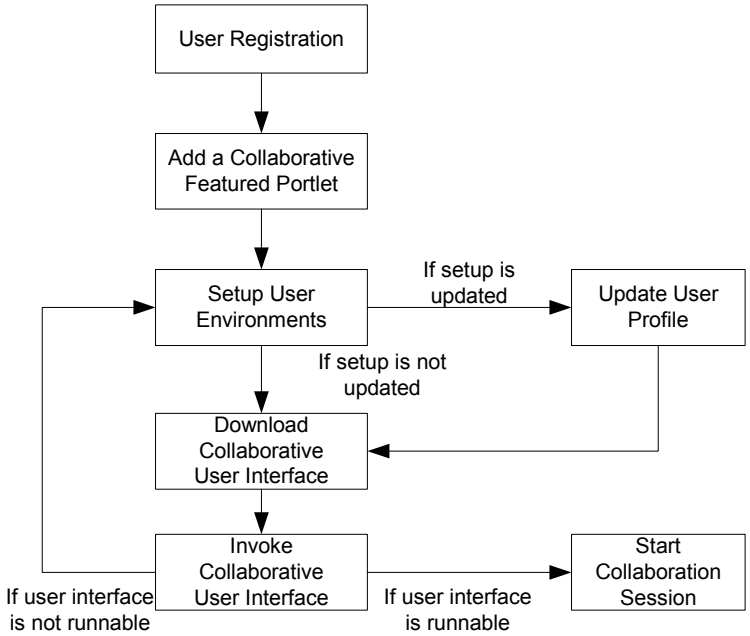
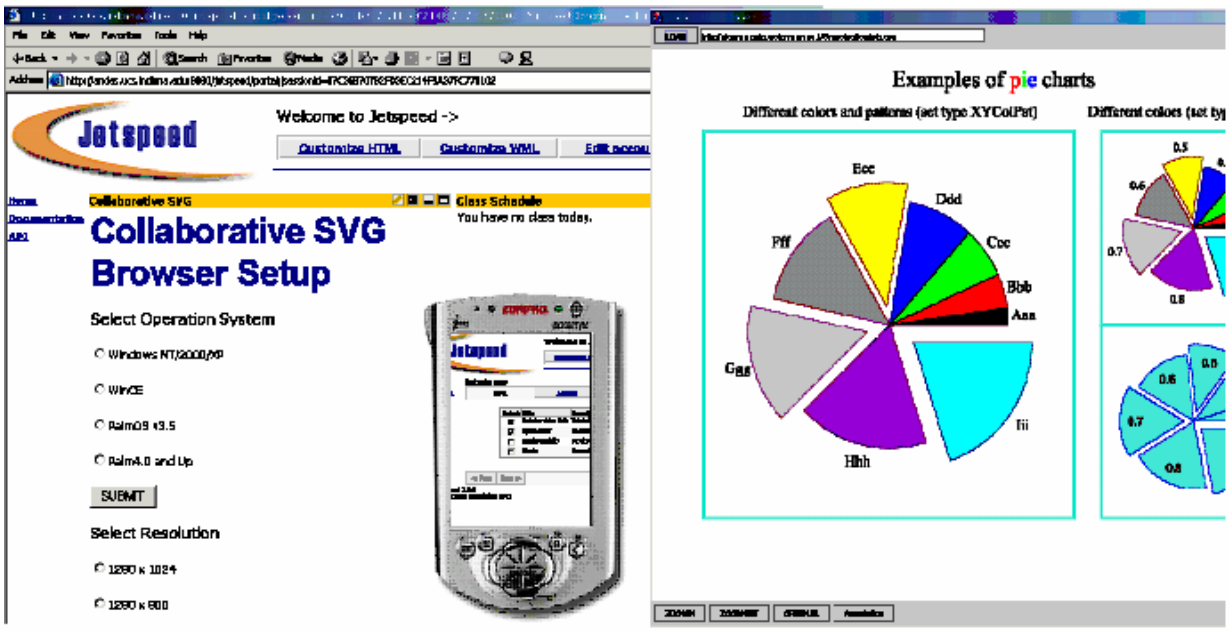


Figure 7.5 Workflow of Setup session in the CAROUSEL Web service

Figure 7.5 illustrates how the Carousel Web service is designed to provide a customizable user environment with its user setup session. First, the user selects the collaborative features in their list of portlets. This is then followed by a setup session where users input their device requirements. Figure 7.6 – (a) is a snapshot of the user display for the setup session in the Carousel Web service. Prior to starting a collaborative session, the users are allowed to change their device profile.

Based on the information in the user device profile, the selected collaborative interface is downloaded. Figure 7.6 – (b) is a snapshot of the collaborative user interface for desktop PC.

Meanwhile, user preferences can be specified during setup sessions, or a user can change some preferences during the session. For example, we allow users to change the resolution of their display in our prototype, or to scroll their display in the prototype of the collaborative SVG browser.



(a) (b)

Figure 7.6 Setup session and collaborative browser

7.3.2 Message filtering for heterogeneous devices

A content server in the CAROUSEL Web Service generates multiple instances of heterogeneous users and collaborative services. We can consider three categories of messages.

- Non-collaborative and collaborative message: Each user can have its own service instance which is not shared with other participants. For instance, each user can browse SVG images based on their preferred resolution by adjusting user preferences. Meanwhile, collaborative events are shared among users. Therefore, those messages should be filtered in the NaradaBrokering according to the purpose of user events.
- Messages for heterogeneous devices: For collaborative messages, different users in different devices will get customized contents generated based on their device profiles. NaradaBrokering filters the specific message and delivers it to the user.
- Messages for the master collaborator and participants: As a policy of collaborative services, we assume that there is a master collaborator and participants. The role of the master collaborator is managed by the session manager, and collaborative messages are assumed only from the users who have the role of master collaborator. There can be multiple master collaborators in a session. NaradaBrokering also filters the user events and deliver to content server.

The filtering in NaradaBrokering is processed based on the architecture of publish and subscribe. When a service is started, the session manager decides the topics that will be used for the sessions and specific devices. Since the session manager is also designed as a Web service, the aggregated portal receives information about the NaradaBrokering node and topic. The aggregated portal distributes the information to users and content server, and the content server starts to process user events and publishes them to be received by specific subscribers of the NaradaBrokering system. For a single

collaborative input, multiple outputs can be published with different topic ids. Meanwhile, each heterogeneous user subscribes to topics within NaradaBrokering based on the information from the session manager. Finally, each user receives customized output generated by the content server, and filtered by the NaradaBrokering system.

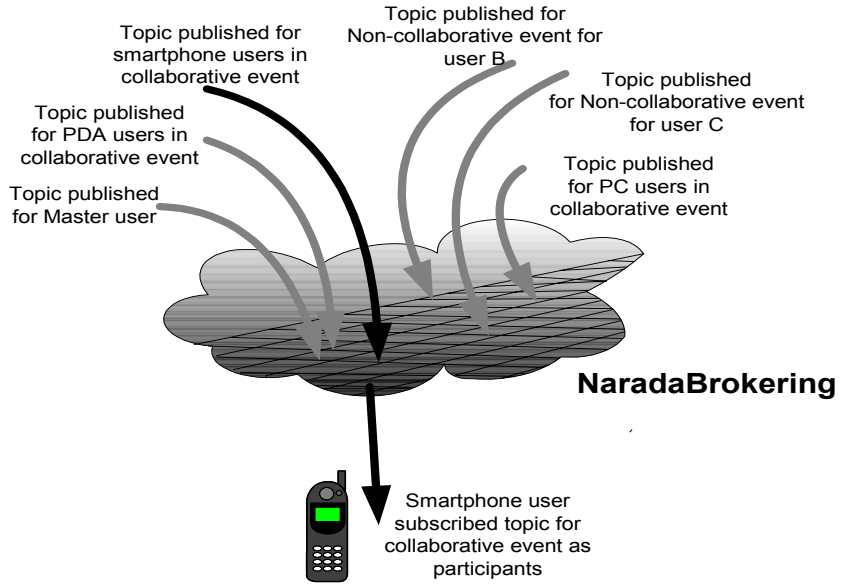


Figure 7.7 Filtering collaborative message for smartphone user in the NaradaBrokering

7.4 Network Communication issues in CAROUSEL Web service (Security, Reliability, Fault Tolerance)

The CAROUSEL Web service provides three phases of network communication: HTTP connection for setup in pre-session, communication with SOAP for invoking

collaborative Web services, and NaradaBrokering connection for real-time collaborative session. Therefore, we have to consider security issues for these network communications.

For the traditional Web Service schema, we can apply general security solutions to the CAROUSEL Web service. We described the basic scheme in Figure 7.8. HTTP connection is used for the communication between users and the aggregative portal web site. Since a user's private information is exchanged via setup and login services, security should be seriously considered in this phase. We can consider the common solution deployed for secure connection such as SSL [Freier+96].

Before starting the real-time collaborative session, the aggregative Web portal should communicate to the back-end Web content server. Between the aggregative Web portal and the content server, SOAP is used as a messaging scheme. We can consider a secure SOAP solution, such as SOAP security extention from W3C [Brown+02].

Eventually, the CAROUSEL Web Service should consider secure communication with NaradaBrokering. Most of the current mobile devices are supported by proxy-based network communication architecture in the transport layer. For instance, wireless LAN, IEEE 802.11b requires a wireless communication access point between wireless PDAs and enables PDAs to access the conventional LAN service. The 3G, and 2.5G network communications also require such a gateway from the wireless service provider. Therefore, ensuring point-to-point network security is feasible in the transport layer, however, the end-to-end network security should be supported by multiple points, such as service providers or local wireless access points.

Security framework provided by NaradaBrokering is based on message level security. Therefore, if mobile devices are capable of encrypting and decrypting using the securely distributed keys, then end-to-end security is able to be ensured in this system. The security framework of NaradaBrokering is described in [Pallickara+03b].

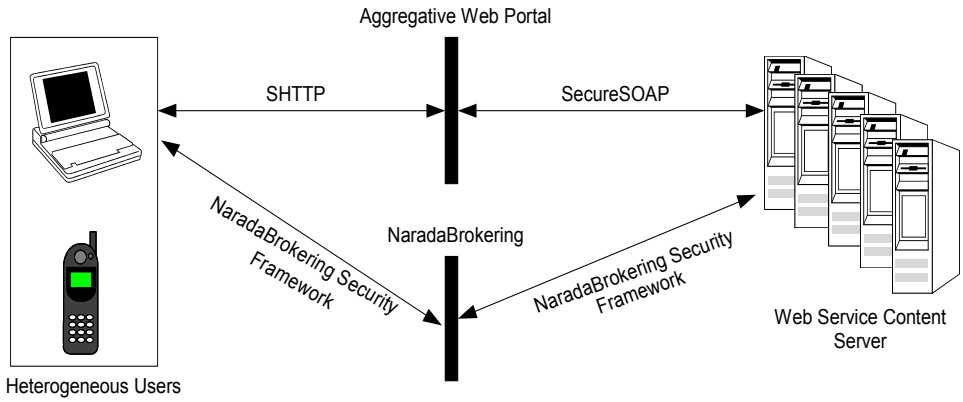


Figure 7.8 Network Communications for the CAROUSEL Web Service

Similarly, some other network communication issues such as reliability and fault tolerances should also be considered in the CAROUSEL Web service. Currently, accessing the aggregative Web Portal is stateless for each client. The state managing in the aggregative Web service is one of the required feature of this architecture.

Meanwhile, NaradaBrokering is JMS compliant [Fox+02c], and messages are delivered based on the JMS specification [Sun+02]. Since HHMS is designed as a network communication protocol supported by NaradaBrokering within the CAROUSEL Web service, it will also inherit the reliability and fault tolerance features of NaradaBrokering. However, wireless networks in mobile computing are is very unreliable. Therefore, a research of reliable network management and fault tolerance are required as advanced research issues.

7.5 Summary

The CAROUSEL Web service is a prototype of collaborative service with MDPA. In this chapter, we described how we deploy our collaboration models within the Web service infrastructure and the architecture of the CAROUSEL Web service.

Our prototype CAROUSEL Web service shows detailed steps of adapting heterogeneous devices. First, user's device profile is adjusted in the setup session. Users are allowed to modify their device profile before starting the collaboration session. Second, during the session, users can change their preferences.

In addition, we discussed security, reliability, and fault tolerance issues for subsequent researches.

CHAPTER 8 CONCLUSION

In this dissertation we have suggested a software architecture for data processing between P2P Grid resources and heterogeneous user devices. This research is driven by the experiences of integrating heterogeneous devices into distributed services of P2P Grids. The users, accessing P2P Grids resources, are already accustomed to utilizing more than one network enabled computing devices, such as laptops, desktops, PDAs, and smartphones.

Our software architecture, MDPA inherits the idea of traditional user interface architectures where there is a separation of user presentation from the original data, and refines data processing as stages of the pipeline for Web-based collaborative application in P2P Grid environments. MDPA provides more adaptable and interoperable architectural environments to the P2P Grid and enables support for heterogeneous user devices with flexible resource accessing strategies. MDPA also supports collaboration between users with a similar flexibility in resource sharing.

Adaptability is an important concept for the design of services accessible from heterogeneous devices. A modular approach of MDPA would facilitate re-use and would only entail small incremental changes to cope with new devices. This thesis presented a modular architecture for designing services in the context of the P2P grid. This modular approach builds upon the emerging standards based approach to building, composing, registering and discovering services.

The impact and scope of this thesis are at three distinct levels. First, it suggests a change in the way services are designed. Second, it outlines a modular approach to this problem which can be expanded incrementally to deal with future changes in the nature of these devices. Finally, though this thesis has been organized in the context of device capabilities, some of the ideas of this thesis could be extended to deal with changing protocol, transport and communication standards.

During the course of this dissertation we have:

- ✓ refined the data process to customize for heterogeneous user environments;
- ✓ categorized the user events based on the pattern of event processing;
- ✓ shown the software architecture based on the Filters and Pipes pattern;
- ✓ modeled the software architecture with Modular PT-net theory;
- ✓ explained the aggregating of services with the pipeline architecture;
- ✓ described how the users can collaborate via the interoperable pipeline architecture;
- ✓ shown how we could deploy the pipeline architecture with a Web service infrastructure;
- ✓ implemented the prototype of a collaborative SVG Web service for heterogeneous environments;
- ✓ explained how MDPA incorporate and interact with content adaptation technologies; and addressed the design problems of adapting content in heterogeneous collaborative services.

8.1 Contributions

In this dissertation we looked to make the following specific contributions to the field of Peer-to-Peer Grids computing:

- *Refinement of data processing*. The data processing in middleware of Peer-to-Peer Grid architecture are refined in MDPA. The data processing is modularized,

and defined with its functional characteristics. This provides a predictable and simplified development environment to interact the P2P Grids.

- *Interoperable data processing architecture.* In order to perform highly interactive distributed services such as collaboration, the software architecture is designed to support interoperability among individual data processing. This dissertation suggests a seamless method of interoperating between the objects.

- *A modeling of collaboration in different degree of flexibilities*. With this software architecture, we also suggest a new aspect of distributed software development supporting a flexible degree of collaboration features. This dissertation deploys the concepts of flexibility in collaborative system design over our software architecture design and suggests collaboration models with different degrees of flexibility in different stages of the MDPA. Thus, it contributes to the systematic design environment of the collaboration system to P2P Grids and also enables the heterogeneous users to have dynamic flexibility in their collaborative features.

- *An adaptable system to support Universal Accessibility.* The data processing to support universal accessibility are processed in various steps. Data filters or transcoders for multimedia information and user profiles are the most popular resources to generate data fitting on the user devices or preferences. The MDPA design enables the P2P Grid to adapt to new emerging technologies and devices more efficiently.

- *Prototype with demonstrative application.* To show our approaches in emerging modern infrastructures, we developed a demonstrative system based on an existing Internet collaboration system. Also for demonstrating collaboration models with various degrees of flexibilities, we built a collaboration system including popular features based on Web service infrastructures. In chapter 3, 4,

and 5, we present our approach with these functional examples, and chapter 6 provides the numeric resultant data to present the performance.

The major contribution of MDPA to the P2P Grid is universality of clients. This is obviously enhances the concept of resources in P2P Grid environments. With MDPA, developers also have advantages with its simplified and systematic data processing environment. MDPA provides generic software architecture for P2P Gird supporting heterogeneous user devices.

8.2 Future works

In the future, other researchers in this field may look to extend this work along a number of different lines.

8.2.1 User Interface

The user interface should consider the flexible accessibility to data object. Each stage of MDPA has different flexibility to access the original data object. The user interface should be designed to consider the multiple flexibilities even in the same service. Since the CPU capability of mobile devices and network bandwidth are improving rapidly, the range of the differences in device capabilities will be far greater than it is now. Therefore, providing multiple flexibilities should be considered as a common design factor of user interface for mobile devices.

8.2.2 Intelligent Messaging Middleware

In the development of collaborative services, the messaging system is a critical factor of system performances. For mobile devices, the messaging system should be accessible from heterogeneous network communication environments seamlessly. As we explained in chapter 5, we develop collaborative Web service based on MDPA with linkage to NaradaBrokering messaging system. Currently we have the message framework, HHMS,

which relays the message from NaradaBrokering messaging system to wireless devices. Since message is delivered to wireless device via PDA adaptor which is a client node of NaradaBrokering system, it causes overload of data transfer.

In addition, developers should reproduce sophisticated features which are provided by NaradaBrokering already. For example, the reliability issues including fault tolerant should be considered in PDA adaptor without any inheritance of advanced features in conventional NaradaBrokering system. Moreover, the information about the users accessed from wireless devices are not revealed to collaborative server transparently.

For improving performance, it should support basic data optimization based on the type of the data. For example, the shared display model of collaboration requires some basic data processing in the message middle ware, such as data compression, or data format conversion.

8.2.3 Deployment with OGSI

Deployment with Open Grid Services Infrastructure (OGSI) [Tuecke+03] is required. The OGSI provides an environments to integrate key Grid technologies with Web service mechanisms to create a distributed system framework. In chapter3, we described that MDPA is deployed within the Web service infrastructure successfully. Since OGSI inherits from the Web service infrastructure, we expect MDPA to be compliant with the OGSI environment as well. OGSI version 1.0 defines a component model that extends WSDL and XML schema definition to incorporate the concepts of stateful Web services, references to instances of services, and asynchronous notification of state change etc. With the features of OGSI, the state management of MDPA can be implemented more easily, such as the lifetime management and multiple instances for collaborative services. Compared to the traditional Web service infrastructure, each stage of MDPA can be implemented more intelligently with OGSI's extended ports types.

8.2.4 Extending Adaptability to Wearable Devices

Heterogeneous devices should be adapted for multiple purposes. Emerging devices are not utilized as mobile clients with limited capability any more. As shown in Figure 8.1, Wearable devices such as Wearable Digital Assistant (WDA), or heads-up display goggles, can be integrated as a part of services with its unique task designed based on the specific advantages inherent in each individual device. For example, the WDA, or wearable camera can assist a presenter in a collaborative session. Meanwhile, in the same session, heads-up display goggles or Intelli-pen can be applied as display or input equipments of participants. The system architecture to integrate these devices working on the wireless network environments such as Bluetooth is also required.

Figure 8.1 Wearable Devices

BIBLIOGRAPHY

[Adler+96] M. Adler, et al."PNG (Portable Network Graphics) Specification Version 1.0", W3C Recommendation, 1996, http://www.w3.org/TR/PNG#Credits

[Adler+01] S. Adler, et el." Extensible Stylesheet Language (XSL) Version 1.0", W3C Recommendation, 2001, http://www.w3.org/TR/xsl/

[ANSI] American National Standards Institute, http://www.ansi.org/

[AOL] America Online Inc. http://www.aol.com

[Apache+99] The Apache XML project, COCOON, http://xml.apache.org/cocoon/

[Arsanjani+02] A. Arsanjani, et al., "Web Service Experience Language Version 2", 2002, http://www-106.ibm.com/developerworks/library/ws-wsxl/

[AT&T] AT&T Laboratories, VNC - Virtual Network Computing from AT&T Laboratories Cambridge, http://www.uk.research.att.com/vnc/

[Barrett+99] R. Barrett and P.P. Maglio, "Intermediaries: An approach to manipulating information streams", IBM Systems Journal, 38, pp.629~641

[Bass+92] L. Bass, R. Faneuf, R. Little, N. Mayer, B. Pellegrino, S. Reed, R. Seacord, S. Sheppard, and M. R. Szczur A Metamodel for the Runtime Architecture of an Interactive System, The UIMS Tool Developers Workshop, SIGCHI Bull., ACM, 24, 1, 1992, pp.32-37

[Bass+98] L. Bass, P. Clements, R. Kazman, Software Architecture in Practice, Addison-Wesley, 1998 (SEI Series in Software Engineering)

[Berners-Lee+97] T. Berners-Lee, "Metadata Architecture", W3C Note, 1997, http://www.w3.org/DesignIssues/Metadata.html

[Berners-Lee+98] T. Berner-Lee, "Uniform Resource Identifiers (URI): Generic Syntax", Draft Standard for URIs, W3C, 1998. http://www.ietf.org/rfc/rfc2396.txt

[Booth+03] D.Booth, M. Champion, C.Ferris, F. McCabe, E. Newcomer, and D. Orchard, "Web Services Architecture", 2003, W3C, http://www.w3c.org/TR/ws-arch

[Bray+00] T. Bray, J.Paoli, C.M. Sperberg-McQueen, and E. Maler, " Extensible Markup Language (XML) 1.0 (Second Edition)", W3C Recommendation, Oct,2000. http://www.w3.org/TR/REC-xml

[Box+00] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte, D. Winer, "Simple Object Access Protocol (SOAP) 1.1", W3C Note 2000, http://www.w3.org/TR/SOAP/

[Brodlie+96] K. Brodlie, J. Wood, and H. Write, "Scientific Visualization: Some Novel Approaches to Learning", in Proceeding of ACM, SIGCSE conference, Barcelona, 1996.

[Brown+01] A. Brown, et al.,"SOAP security extentions: Digital Signature", W3C Note, 2001, http://www.w3.org/TR/SOAP-dsig/

[Chandra+00] S. Chandra, C. Ellis and A. Vahdat, "Differentiated Multimedia Web Services using Quality Aware Transcoding", InfoCom, 9th Annual Joint Conference Of The {IEEE} Computer And Communications Societies, 2000

[Chandra+01] S. Chandra, A. Gehani, C. Ellis and A. Vahdat, Transcoding Characteristics of Web Images, Multimedia Computing and Networking, Jan,

2001, San Jose, CA.

[Chisholm+99] W. Chisholm, G. Vanderheiden, and I. Jacobs, “Web Content Accessibility Guidelines 1.0”, W3C Recommendation, 1999, http://www.w3.org/TR/WCAG10/

[Christensen+01] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, "Web Services Description Language (WSDL) 1.1", W3C Note, 2001, http://www.w3c.org/TR/wsdl

[Coulouris+01] G. Coulouris, J. Dollimore, and T. Kindberg, Distributed Systems Concepts and Design, Addison Wesley, Third Edition, 2001

[Coutaz+87]J. Coutaz, “PAC, an Object Oriented Model for Dialog Design”, in Proceedings Interact’87 (North Holland, 1987), pp.431-436

[Coutaz+95] J. Coutaz, L. Nigay, D. Salber, "Agent-Based Architecture Modelling for Interactive Systems", in: P. Palanque, D. Benyon (eds.), Critical Issues in User Interface Engineering, Springer-Verlag, London, 1995, pp. 191-209

[Coutaz+97] J. Coutaz, “PAC-ing the Architecture of Your User Interface”, in Proceedings Eurographics Workshop on Design, Specification and Verification of Interactive Systems, Springer Verlag.,1997, pp. 15-32

[Chu+98]Y. Chu, P. DesAutels, B. LaMacchia, and P. Lipp, “PICS Signed Labels(DSig) 1.0 Specification”, W3C Recommendation, 1998, http://www.w3.org/TR/REC-DSig-label/

[DCMI] Dublin Core Metadata Initiative, http://dublincore.org/

[Dewan+95] P. Dewan, “Multiuser Architectures”, in Proceedings EHCI’95, Working Conference on Engineering Computer Human Interaction.

[Drath] Rainer Drath, “Visual Object Net++”, http://www.systemtechnik.tu-ilmenau.de/~drath/visual_E.htm

[eScience] United Kingdom e-Sciene Activity, http://www.escience-grid.org.uk

[ETSI] European Telecommunications Standard Institute, http://www.etsi.org/

[Freier+96] A. Freier,P.Karlton, P.Kocher, “The SSL Protocol, Ver. 3.0”, *http://wp.netscape.com/eng/ssl3/ssl-toc.html*

[Foster+98] I. Foster and C. Kesselman (Eds.), "The Grid: Blueprint for a New Computing Infrastructure", Morgan-Kaufman, 1998. See especially D. Gannon, and A. Grimshaw, "Object-Based Approaches", pp. 205-236, of this book.

[Fox+98] A. Fox, S. Gribble, Y. Chawathe and E. A. Brewer, “Adapting to Network and Client Variation Using Active Proxies: Lessons and Perspectives,” in A special issue of IEEE Personal Communications on Adaptation, 1998, http://citeseer.nj.nec.com/fox98adapting.html

[Fox+02a] Geoffrey Fox, Dennis Gannon, Sung-Hoon Ko, Sangmi Lee, Shrideep Pallickara, Marlon Pierce, Xiahong Qiu, Xi Rao, Ahmet Uyar, Minjun Wang, Wenjun Wu, Book chapter on Peer-to-Peer Grids http://grids.ucs.indiana.edu/ptliupages/publications/p2pgridbook.pdf

[Fox+02b] Geoffrey Fox, Sung-Hoon Ko, Kangseok Kim, Sangyoon Oh, Sangmi Lee on "Integration of Hand-Held Devices into Collaborative Environments" IC'02 June 2002 Las Vegas, NV, http://grids.ucs.indiana.edu/ptliupages/projects/carousel/papers/PDA_IC2002.pdf

[Fox+02c] Geoffrey Fox, Shrideep Pallickara, “JMS Compliance in the Narada Event Brokering System”, Proceedings of the 2002 International Conference on Internet Computing (IC-02). Volume 2 pp 391-397

[Fox+02d] Geoffrey Fox, Sangmi Lee, Sung-Hoon Ko, Kangseok Kim, Sangyoon Oh, “Carousel Web service: Universal Accessible Web service Architecture for Collaborative Application”, http://grids.ucs.indiana.edu/ptliupages/projects/carousel/papers/Carousel_PerCom03.pdf

[Fusion] DoE Fusion Grid at http://www.fusiongrid.org

[Globus] Globus Grid Project http://www.globus.org

[Goldberg+84] A. Goldberg, “Smalltalk-80: The Interactive Programming Environment, Addison-Wesley Publ., 1984.

[Grid] The Grid Forum http://www.gridforum.org

[GridWG] GridForum Grid Computing Environment working group(http://www.computingportals.org) and survey of existing grid portal projects. http:www.computingportals.org/cbp.html

[Harakan] Harakan Software, PalmVNC,http://www.btinternet.com/~harakan/PalmVNC/

[Hendler+02] J. Hendler, T. Berners-Lee, E. Miller, " Integrating Appliations on the Semantic Web," Journal of the Institute of Electrical Engineers of Japan, Vol 122(10), October, 2002, p. 676-680, http://www.w3.org/2002/07/swint

[Hoschek+02] W. Hoschek, "The Web Service Discovery Architecture",In Proc. of the Int'l. IEEE/ACM Supercomputing Conference (SC2002), 2002

[Huffman+52] Huffmann, D.," A Method for the Construction of Minimum Redundancy Codes," Proceedings of the IRE 40(9): 1098-1101, 1952

[IBM+02a] IBM Raleigh Lab., "Transcoding Technology in WebSphere Everyplace Access: Using Transcoding Technology to Expand your Pervasive Portal", IBM WebSphere Developer Technical Journal, Sept, 2002. http://www7b.boulder.ibm.com/wsdd/techjournal/0209_thrasher/thrasher.html

[IBM+02b] IBM Research, "Open Visualization Data Explorer ", 2002, http://www.research.ibm.com/dx/

[IDEAlliance] International Digital Enterprise Alliance, http://www.idealliance.org/

[Intel] Intel Inc., Intel quickweb, http://www-us-east.intel.com/quickweb/

[IVDGL] International Virtual Data Grid Laboratory, http://www.ivdgl.org/

[Jacobs+99] I. Jacobs, J. Brewer, "Accessibility Features of CSS", W3C Note, 1999, http://www.w3.org/TR/CSS-access

[JetSpeed] The Apache Jakarta Project, Jetspeed, http://jakarta.apache.org/jetspeed/site/features.html

[Johnston+00] W. Johnston, D. Gannon, B. Nitzberg, A. Woo, B. Thigpen, L. Tanner, "Computing and Data Grids for Science and Engineering," Proceedings of Super Computing 2000.

[Kay+03] M. Kay, "XSL Transformation (XSLT) Version 2.0", W3C Working Draft 2003, http://www.w3.org/TR/xslt20/

[Koivunen+99] M. Koivunen, "Accessibility Features of SMIL", W3C Note, 1999, http://www.w3.org/TR/SMIL-access/

[LAAS] Laboratory for Analysis and Architecture of Systems, "Tina Ver. 2.5.1", http://www.laas.fr/tina/announce-2.5.1.html

[Lassila+99] O. Lassila, and R.R.Swick, "Resource Description Framework(RDF) Model and Syntax Specification", W3C Recommendation, 1999, http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/

[Lee+02] Sangmi Lee, Geoffrey Fox , Sunghoon Ko, Minjun Wang, Xiaohong Qiu "Ubiquitous Access for Collaborative Information System Using SVG", Proceedings of SVG OPEN, July, 2002, Zurich, Switzerland

[Lee+03a] Sangmi Lee, Sunghoon Ko, Geoffrey Fox, Kangseok Kim, Sangyoon Oh, "A Web Service Approach to Universal Accessibility in Collaboration Services" in Proceedings of 1st International Conference on Web Services Las Vegas June 2003

[Lee+03b] Sangmi Lee, Sunghoon Ko, Geoffrey Fox, Adapting Content for Mobile Devices in Heterogeneous Collaboration Environments in Proceedings of the 2003 International Conference on Wireless Networks Las Vegas June 2003

[Maheshwari+01] A. Maheshwari, A. Sharma, K. Ramamritham and P. Shenoy, "TranSqui: Transcoding and Caching Proxies for Heterogeneous E-Commerce Environments", Technical Report, 2001, http://www.cs.umass.edu/Dienst/UI/2.0/Describe/ncstrl.umassa_cs%2FUM-CS-2001-051

[MSN] Microsoft Corporation http://www.msn.com

[MSAF] Multimedia Services Affiliate Forum, http://www.msaf.org/

[McCathieNevil+00] C. McCathieNevile, M. Koivunen, "Accessibility Features of SVG", W3C Note, 2000, http://www.w3.org/TR/SVG-access/

[Mohan+99] R. Mohan, J. Smith and C. Li, "Adapting Multimedia Internet Content for Universal Access", IEEE Transaction on Multimedia, vol.1, pp.104-114, 1999.

[NEES] NEES Earthquake Engineering Grid, http://www.neesgrid.org/

[Nokia] Nokia, Java VNC viewer, http://www.mgroeber.de/nokia.htm

[OASIS+03] OASIS, "OASIS Technical Committee Guidelines", 2003, http://www.oasis-open.org/committees/guidelines.php

[OpenOffice] OpenOffice.org , OpenOffice. http://www.openoffice.org

[Oracle] Oracle Inc., "Oracle portal-to-go; any service to any device", white paper, Oct,1999, http://www.oracle.com/mobile/portaltogo/

[Oram+01] Oram, A. (eds) 2001. Peer-To-Peer: Harnessing the Power of Disruptive Technologies, March, 2001, O'Reilly, ISBN 0-596-00110-X

[Palm+02] Palm Inc. "Web Clipping Apps Development", 2002, http://www.palmos.com/dev/tech/webclipping/

[Pallickara+03a] Shrideep Pallickara, Geoffrey Fox, "NaradaBrokering: A Middleware Framework and Architecture for Enabling Durable Peer-to-Peer Grids", in proceedings of ACM/IFIP/USENIX International Middleware Conference, 2003, http://www.naradabrokering.org/papers/NB-Framework.pdf

[Pallickara+03b] Shrideep Pallickara, Marlon Pierce, Geoffrey Fox, Yan Yan, Yi Huang, " A Security Framework for Distributed Brokering Systems", http://www.naradabrokering.org/papers/NB-SecurityFramework.pdf

[Phaff+85] G.E. Phaff, et al.," User Interface Management System", Eurographics Seminars, Springer verlag, 1985

[PhysicsGrid] Particle Physics Grid Project Site, http://www.griphyn.org/

[Probets+00] S. Probets, “Flash and SVG”, http://broadway.cs.nott.ac.uk/projects/SVG/flash2svg/

[Rao+01] C. Rao, D, Chang, Y, Chen and M, Chen, “iMobile: A Proxy-Based Platform for Mobile Services”, Wireless Mobile Internet, 2001, pp.3-10

[Reynolds+00] F. Reynolds, C. Woodrow, H. Ohto, “Composite Capability/Preference Profile(CC/PP) Structure”, W3C working draft, July, 2000, http://www.w3.org/TR/2000/WD-CCPP-struct-20000721/

[Scallan] T.Scallan, "CORBA Primer Whitepaper", Whitepaper, Segue Software, Inc. http://www.omg.org/news/whitepapers/seguecorba.pdf

[SCEC] SCEC Earthquake Science Grid, http://www.scec.org

[Shaeck+02] T. Schaeck, “ Web Services for Remote Portals (WSRP) Whitepaper”, 2002, http://www.oasis-open.org/committees/wsrp/documents/wsrp_wp_09_22_2002.pdf

[Smith+99]J.R. Smith, R. Mohan, and C.Li, “Scalable multimedia delivery for pervasive computing.” In Proc. of ACM Multimedia, Oct. 1999, pp.131-140, ACM Multimedia, Orlando, Florida

[Storer+82] J. Storer, and T.G. Szymanski, “Data Compression via Textual Substitution”, Journal of the ACM 29:928-951, 1982

[SUN] Sun Microsystems, Inc. "JAVA 2 Platform", http://www.java.sun.com/

[SUN+02] Sun Microsystems Inc. “Java Message Service”, http://java.sun.com/products/jms/docs.html

[Sun+03] Sun Microsystems Inc., "Project JXTA Technology: Creating Connected Communities ", 2003, http://www.jxta.org/project/www/docs/JXTA-Exec-Brief-032803.pdf

[TEI] Text Encoding Initiative, http://www.tei-c.org/

[Tuecke+03] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maquire, T. Sandholm, D. Snelling, and P. Vanderbilt, "Open Grid Services Infrastructure (OGSI) Version 1.0 (draft)", http://www.ggf.org/ogsi-wg

[UDDI+00] Universal Description, Discovery and Integration(UDDI), "UDDI Technical White Paper", 2000, http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf

[UnicodeConsortium] The Unicode Consortium, http://www.unicode.org/unicode/consortium/consort.html

[Upsen+89] Upson, C., Faulhaber Jr, T., Kamins, D., Laidlaw, D., Schlegel, D., Vroom, J., Gurwitz, R., et al. "The Application Visualization System: A Computational Environment for Scientific Visualization". IEEE Computer Graphics & Applications,1989

[Wood+95] J. Wood, H. Write, and K. Brodlie, "CSCV – Computer Supported Collaborative Visualization", in Proceedings of Display Group International Conference on Visualization and Modeling, 1995

[Wang+00] H. Wang, B. Raman, C.Chuah, R. Biswas, R. Gummadi, B. Hohlt, X. Hong, E. Kiciman, Z. Mao, J. Shih, L. Subramanian, B. Zhao, A. Joseph, and R. Katz, ICEBERG: An Internet-core Network Architecture for Integrated Communications, IEEE Personal Communications, Special Issue on IP-based Mobile Telecommunication Networks, 2000

[WebSphere] IBM Inc. "WebSphere software platform", http://www-3.ibm.com/software/info1/websphere/index.jsp?tab=highlights

[Yahoo] Yahoo! Inc., http://www.yahoo.com

[Yeong+95] W. Yeong, T. Howes, S. Kille, " Lightweight Directory Access Protocol", Internet Request for Comments:1777,Standards Track, Network Working Group, http://www.umich.edu/~dirsvcs/ldap/doc/rfc/rfc1777.txt

BIOGRAPHICAL SKETCH

Name Sangmi Lee

Date of Birth April,28,1970

Place of Birth Seoul, Korea (ROK)

Education

- M.S. Computer and Information Science, Syracuse University, NY, USA, 2000
- B.S. from Sookmyung Women's University in Physics, Seoul, Korea, 1993

Publications

- **Journal Articles**

Geoffrey Fox, Sung-Hoon Ko, Marlon Pierce, Ozgur Balsoy, Jake Kim, Sangmi Lee, Kangseok Kim, Sangyoon Oh, Xi Rao, Mustafa Varank, Hasan bulut, Gurhan Gunduz, Xiaohong Qui, Shrideep Pallickara Ahmet Uyar, Choonhan Yoon. "Grid Services for Earthquake Science", *Concurrency and Computation: Practice and Experience* in ACES Special Issue, 2002.

- **Conference Articles**

Sangmi Lee, Sunghoon Ko, Geoffrey Fox, Kangseok Kim, Sangyoon Oh, "A Web Service Approach to Universal Accessibility in Collaboration Services" in Proceedings of *1st International Conference on Web Services*, Las Vegas, June 2003

Sangmi Lee, Sunghoon Ko, Geoffrey Fox, "Adapting Content for Mobile Devices in Heterogeneous Collaboration Environments", in Proceedings of *the International Conference on Wireless Networks,* Las Vegas, June 2003

Hasan Bulut, Geoffrey Fox, Dennis Gannon, Kangseok Kim, Sung-Hoon Ko, Sangmi Lee, Sangyoon Oh, Xi Rao, Shrideep Pallickara, Quinlin Pei, Marlon Pierce, Aleksander Slominski, Ahmet Uyar, Wenjun Wu, Choonhan Youn , "An Architecture for e-Science and its Implications" to be presented at *2002 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2002)* July 17 2002

Sangmi Lee, Geoffrey Fox, Sunghoon Ko, Minjun Wang, Xiaohong Qui, "Ubiquitous Access for Collaborative Information System Using SVG", *SVG Open*, 2002, July, 2002, Switzerland

Geoffrey Fox, Sung-Hoon Ko, Kangseok Kim, Sangyoon Oh, Sangmi Lee, "Integration of Hand-Held Devices into Collaborative Environments", in Proceedings of *the International Conference on the Internet Computing*, June 2002

- **Book Contribution**

Geoffrey Fox, Dennis Gannon, Sung-Hoon Ko, Sangmi Lee, Shrideep Pallickara, Marlon Pierce, Xiaohong Qiu, Xi Rao, Ahmet Uyar, Minjun Wang, Wenjun Wu Book chapter on Peer-to-Peer Grids. Making the Global Infrastructure a Reality Grid. Published by John Wiley, West Sussex, England. ISBN 0-470-85319-0, 2003

- **Technical Reports**

Geoffrey Fox, Sangmi Lee, Sunghoon Ko, Kangseok Kim, Sangyoon Oh, "CAROUSEL Web service: Universal Accessible Web service Architecture for Collaborative Application", Community Grid Lab,

Indiana University, November 2002

Geoffrey Fox, Sung-Hoon Ko, Kangseok Kim, Sangyoon Oh, Sangmi Lee “Integration of Hand-Held Devices into Collaborative Environments”, Community Grid Lab, Indiana University, January 2002