International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 2 Issue 12, December - 2013

A Survey on Distributed Multimodal System Using Suitable Design Patterns

Ashish N. Patil[1], Manjusha Joshi[1], Dr. S. D. Joshi[2], Dr. R. M Jalnekar[3]

[1]*Research Students, BVDUCOE, Pune, Maharashtra, India*

[2]*Professor, BVDUCOE, Pune, Maharashtra, India*

[3]*Director, VIT, Pune, Maharashtra, India*

Abstract

Distributed System Architecture (DSA) is the ideal solution for integrating processes when there are multiple units, control rooms or geographically distributed locations. With Distributed System Architecture, users experience a single, totally integrated system instead of several independent systems, while retaining the ability to autonomously manage each system. Hence it provides optimum functionality and flexibility. Multimodal interactions and multimodality refer to the process in which different devices and people are able to interact aurally, visually, by touch or by gesture. Ubiquitous computing refers to this interaction happening anywhere, anytime, using any device, often in order to increase the accessibility and improve the user experience. Modelling multimodal interaction is no simple task, due to the multiple input and output channels, modes and the combination of possibilities between data coming from different sources, not to mention output modality selection based on context and user profile. The huge challenge of multimodal interface creation is to build reliable processing in the multimodal system to analyze and to understand multiple communications means in real-time. Our aim to reduce the complexity associated with building multi-modal systems; hence there is need to build an adaptable multi-agent architecture for multimodal systems and provide a suite of general purpose, plug-in agents to handle essential tasks like speech I/O, fusion and semantic analysis. And also there is need to specify a suitable pattern to create a platform which is able to provide multimodal interactions between people with functional diversity and several semantic services developed by third-party companies and consumed through a presentation, in order to make content more accessible and interactive in distributed environment. The general purpose of this research is the development and deployment of urban, ubiquitous services in efficient manner for people with distributed system (e.g. Internet).

Designing multimodal approach, along with the traditional computing environment or with the ubiquitous environment, is always a challenging task in the HCI field. The key idea of this paper to study an adaptable architectural pattern for multi-modal interaction for configurable distributed systems and make the multimodal interface design more transparent, flexible, efficient, and powerfully expressive means of human-computer interaction (HCI).

Keywords: ***DSA*** *(Distributed System Architecture),* ***HCI*** *(Human-Computer Interaction)*

1. Introduction

In today's world practice most interactions with computers take place using multiple I/O modalities. Input modalities are event-based or streaming based and it requires a user device to transfer human output into a form suitable for computer processing. Event-based input modalities—such as input via a keyboard or mouse—react to user actions by producing discrete events. Streaming-based modalities sample input signals with some resolution and frequency, producing a time-stamped array of sampled values. For example, a computer detects a user's voice or psychological signals by sampling input signals with sensors such as a microphone. The interaction and the interface between

humans and computers are not very natural yet; although they are often lightly multimodal (considering, for instance, the mouse or other pointing devices alongside the keyboard). The appearance of touch screens improved user's navigation through the graphical user interface, but did not introduce a real multimodal interaction between users and computers. Multimodal interactions and multimodality refer to the process in which different devices and people are able to interact aurally, visually, by touch or by gesture. Ubiquitous computing refers to increase the accessibility and improve the user experience.

Multimodality is applied to present usable and accessible information to users, including videos, images and texts. One of the main purposes of multimodality is the improvement of user interactions with devices, such as smart-phones. Information access and interaction for users with functional diversity, is another main purpose of multimodality. User interfaces should allow users to interact with the content or a service through the most appropriate modes, taking into account user's preferences and context. Interactions can be defined as information exchange among people, technologies (represented by user interfaces) and processes. The user may determine the mode or modes of interaction that he prefers for accessing information more naturally. Multimodality extends and improves user interfaces because it allows the integration of voice, image and other types of data input such as keyboards, mice, webcams, pens, touch screens and remotes. The motivating scenario behind the design and development of the multimodal platform and semantic services is one in which people use media poles and confidently interoperate with the surrounding environment in a multimodal way.

Multimodal systems are computer systems empowered with multimodal capabilities for human/machine interaction and able to interpret information from various sensory and communication channels. Literally, multimodal interaction offers a set of "modalities" to users to allow them to interact with the machine. Multimodal interfaces process two or more combined user input modes (such as speech, pen, touch, manual gesture, gaze, and head and body movements) in a coordinated manner with multimedia system output. There are a new class of interfaces that aim to recognize naturally occurring forms of human language and behaviour, and which incorporate one or more recognition-based technologies (e.g. speech, pen, vision). Two unique features of multimodal architectures and processing are: (1) the fusion of different types of data; and (2) real-time processing and temporal constraints imposed on information processing.

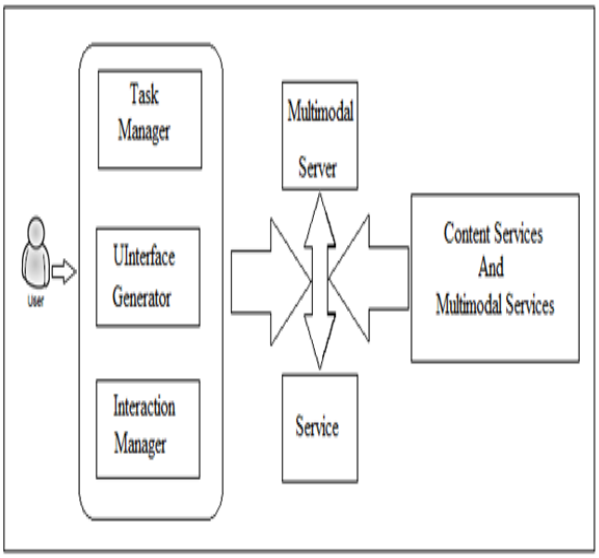

Figure 1. Architecture of Distributed Multi-modal System

1.1 Architecture of distributed multi-modal system -

A model-view controller (MVC) pattern managing all data and interactions among the users, the system and the external services.

Task Manager (TM) - The main goal of the Task Manager (TM) is to manage multiple users. The TM has all the user profiles serialized, and knows the context and the user's preferences (which are managed by the User Modeller) in order to adapt the system and the content according to user's necessities and context.

Interaction Manager (IM) – The IM is the controller into a MVC pattern and it is the core of the architecture and its role is the management of the user interactions. The TM creates a task, generating an instance of the user's context in order to inform the IM about the user's preferences, and the IM adapts the content and integrates it with multimodality for the user. The TM is able to get the content of third-party distributed services in order for the IM to generate the user interface. The content will be obtained through the Service Broker which would manage Semantic Web Services.

Multimodal Server (MS) - As a MVC pattern component, the TM is the model, because it interacts with the IM, making transparent all the accesses to

services, the message transformation, and the protocol used by the MS. The User Modeller is another component of the model. The user interface is generated with the User Interface Generator (UI Generator) module, which would be the view. The MS is included into the controller modules with the IM and it is able to manage all third-party distributed services in order to process and synthesize human-computer interactions. The main goal of this module is to provide communication, in the experimental platform, among the interface and the multimodal services able to understand the user and communicate with user.

Multimodal services - Multimodal services are managed directly by the MS. Distributed, non-local services currently available in the experimental platform For example, if the user is interacting with the system using the voice, the voice recognition process will know how to connect with the user's device because the MS sends it a message with the IP address, port and other security parameters. All messages among all services and system processes are exchanged using extensible mark-up language based language.

Content services - Content services are third-party, semantic, Web services (e.g. city services, weather services) managed through the Service Broker, which will be in charge of discovering and requesting new services, and adapting the content depending on the context, user preferences and IM requests.

1.2 Applications of multi-modal system in distributed Environment –

Multimodal interfaces offers flexibility providing multiple channels for interaction. The main idea of the experimental platform developed is to manage distributed services that offer the capability of processing and synthesizing the multiple modalities of interaction of the user interface.

These services are enriched with semantics and adapted to user necessities through an adaptor and a service bus which manage the multimodal, distributed, semantic services depending on user's preferences, necessities and modelling. For example, an online meeting is a meeting in which the presenter connects to the attendees through an application in distributed environment. This option allows the presenter and attendees to share video, audio, programs, instant messages and a shared desktop. So in this system, information and communication technology systems suffer from an inability to satisfy the heterogeneous needs of many users.

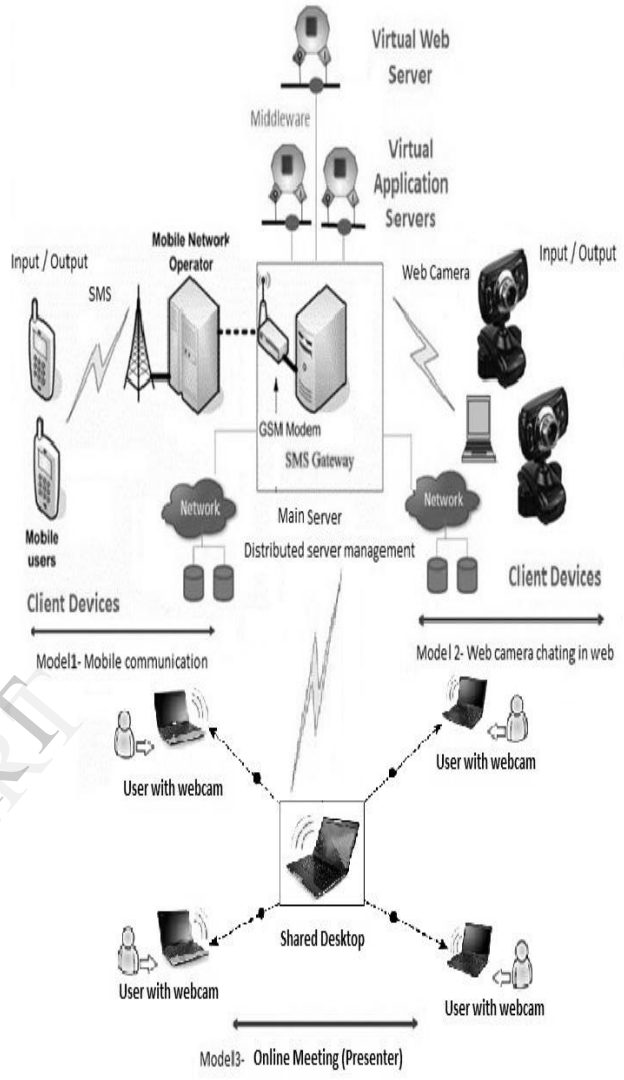

Figure 2. Example of Distributed multi-modal system

2. Background & Motivation

2.1 What is distributed system? –

Distributed systems are collection of independent computers or group of networked computers which have the same objective for their function. This kind of system has the following characteristics -

1) **Connecting resources and users is a means of resource sharing and load balancing –** It provides efficient and responsive resource utilization.

2) **Distribution transparency** - It is to make the existence of multiple computers invisible (transparent) and provide a single system image to its users.
3) **Scalability** - It refers to the capability of a system to adapt to increased service load.
4) **Reliability, availability and fault tolerance** - It achieved through redundancy and dynamic allocation.
5) **Enhanced performance potential** - It derived from parallel operation in huge distributed system.

Distributed system is classified into two types- 1) Homogeneous distributed system 2) Heterogeneous distributed system.

In homogeneous DS, all sites they have identical software as well as hardware configuration and it appears to user as single system. All sites are known to each other so that they agree to co-operate with user request. In heterogeneous DS, different sites use different schemas of hardware, platform and language. So it may not be aware of each other & they provide only limited facilities for co-operation in transaction processing.

2.2 Architecture of Distributed System –

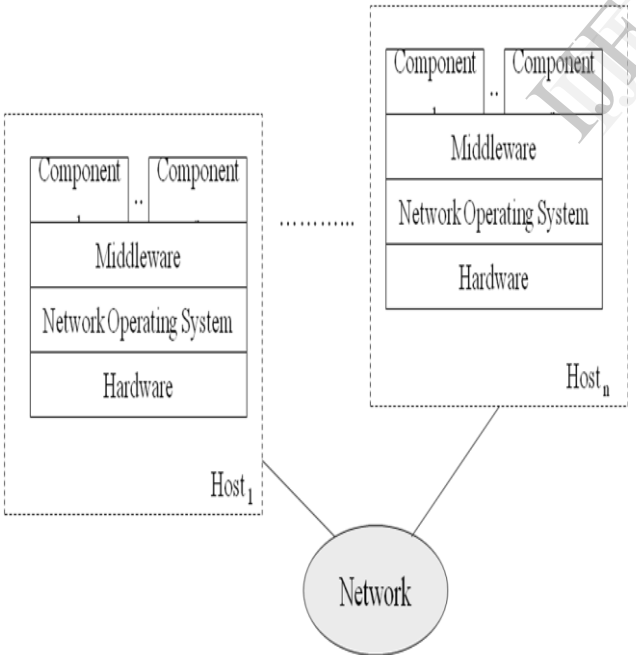

Figure 3. Distributed System Architecture

Various hardware and software architectures are used for distributed computing. At a lower level, it is necessary to interconnect multiple CPUs with some sort of network, regardless of whether that network is made up of loosely-coupled devices and cables. At a higher level, it is necessary to interconnect processes running on those CPUs with some sort of communication system. Distributed programming typically falls into several basic architectures viz. Client-server, 3-tier architecture, N-tier architecture, distributed objects, loose coupling or tight coupling.

2.3 Components of distributed system –

a. Computer workstations (sites or nodes) - A distributed DBMS consists of a number of computer workstations that form the network system. The distributed database system must be independent of the computer system hardware and situated across the nodes which are participants in network.

b. Network components (both hardware and software) – Each workstation in a distributed system contains a number of network hardware and software components. These components allow each site to interact and exchange data with each other site. Network system independence is a desirable property of the distributed system.

c. Communication media – In a distributed system, any type of communication (data transfer, information exchange) among nodes is carried out through communication media. This is a very important component in a distributed management system. It is desirable that a distributed DBMS be communication media independent, that is, it must be able to support different types of communication media (e.g. wired or wireless communication).

d. Middleware – Too many networked applications existed hard or difficult to integrate due to departments are running on different platforms (NOSs). So middleware plays an important role in integration and interoperability only at primitive level NOS services.

e. Transaction Manager (TM) – A TM is a software component that resides in each computer connected with the distributed system and is responsible for receiving and processing both local and remote applications data requests.

f. Data Manager (DM) – A DM is also a software component that resides in each computer connected with the distributed system and stores and retrieves data located at that site. In a distributed DBMS, a DM may be a centralized DBMS.

2.4 Demand of distributed system –

A distributed system may have a common goal such as solving a large computational problem as well as connecting resources to improve its utilization. There are genuine benefits in building distributed systems are.

1) **Distributing machines may make resource to another** - Each computer may have its own user with individual needs and the purpose of the distributed system is to coordinate the use of shared resources or provide communication services to the users.
2) **Co-operative and social networking** - Clients that are geographically differentiated can now work and play together. Examples of this are plenty - distributed document systems, remote desktop sharing or handling PC, audio or video conferencing, messaging, multiplayer games and social networks.
3) **Increased reliability** - Assuming that a small percentage of machines break; whatever remains of the framework are done through another participants in that framework and also can do useful work. Hence the system removes failures in individual computer system in network.
4) **Incremental growth** - It has delivery aided capacity when and where needed. The structure of the system (network topology, network latency, number of computers) is not known in advance, the system may consist of different kinds of computers, network links and the system may change during the execution of a distributed program. Networking allows permits you to add onto an existing infrastructure.
5) **Remote access and services** - Users may need to access information held by others at their systems. Examples of this include web browsing, remote file access, and programs to retrieve large files.
6) **Mobility** - Clients move around with their PCs and cell phones. It is not feasible for them to carry all the qualified information they require with them.

There are also some suitable design patterns for performance modelling of distributed management system.

2.5 Design Patterns for modelling of distributed multimodal system –

In a distributed application system, in many cases the complex between objects are many communication relations, and therefore in the development process we utilize design patterns on the complementary advantages of each phase and reward each other to enhance the overall stability of the system architecture and robustness. Design patterns are recurring solutions to software design problems you find again and again in real-world application development. Patterns are about design and interaction of objects, as well as providing a communication platform concerning refined, reusable solutions to commonly encountered programming challenges [3].

a. Observer & mediator design pattern - It defines a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically. This allows keeping a single copy of the data and having multiple other objects depend on them. Mediator and Observer are competing patterns. The difference between them is that Observer distributes communication by introducing Observer and Subject objects, whereas a Mediator object encapsulates the communication between other objects. In a distributed systems development, the server and server interaction must be to maintain this constraint by the Observer and the Subject object between the client and server-side or between the client and the client, thus a large number of distributed objects can communicate. Mediator Controller is responsible for maintaining transparent communication between objects through an intermediary object to encapsulate a series of object interaction, and mutual cooperation does not need to explicitly reference the object, thereby reducing coupling between objects, hence the interaction between objects is a kind of weak dependence relationship which makes each object independently, So it is beneficial to enhance reusability and maintainability of the system that reflects the one to one communication relationships[5].

b. Model Driven Architectural (MDA) Pattern - The model driven architecture proposes sophisticated specification in order to develop a distributed system in rapid ways. MDA provides a wide variety of specifications in order to develop a system as a whole entity. Unified Modelling Language (UML) in MDA ensures the development of distributed system with proper requirement specification that means it is a standardized specification language for object-oriented mode-ling. MDA development process uses both platform-independent model (PIM) and platform-specific model (PSM) which has a high-level abstraction and is independent of any implementation technologies. It represents the business functionality of the complete system which is to be developed. PSM

contains information about the syntactic, semantic, and presentation information in the form of a UML. UML based performance modelling techniques provide the means for modelling complex distributed systems. The key issue in this technique is the use of abstractions for separating application level issues from the use of technical resources (e.g. network adapters and network latencies). Hence Model-driven architecture separates business logics from underlying platform technologies. It provides tools to specify a system independently of its supporting platform. [7]

c. Service Oriented Architectural (SOA) Pattern - Service-oriented architectural businesses and their processes are implemented through Web services and their interactions. Web services interactions can be distributed across applications and enterprises. An e-Business solution therefore requires the platform to support service interaction by providing a number of common integration services. The business model describes both the pattern and protocol of exchanging messages between the services. Service-Oriented Architecture is envisioned most efficient in designing enterprise systems and integrating heterogeneous applications in a distributed environment as well as it enables complex systems to be divided into loosely coupled services. The most essential part of SOA is that it differentiates the service's implementation from its interface. So Web services technologies make system components to be interoperable and extensible within an enterprise as well as across enterprise boundaries. [1][8]

d. Multi-agent Architectural Pattern - In today's world practice, distributed systems are increasingly complex, often distributed over several sites and consist of software interacting with each other or with humans. So there is need to identify and analyze all system problems to find models for multi-agents to implement and integrate them into a coherent system.

Agent technology is a software prototype that permits to implement large and complex distributed applications in distributed environment. An agent is a computer system within an environment and with an autonomous behaviour, made for achieving the objectives that were set during its design. A multi-agents system is a system that contains a set of agents that interact with communications protocols and are able to act on their environment. Agent-orientation provides a level of abstraction above that of object orientation and is supported by several platforms, frameworks, languages and methodologies. An adaptable multi-agent architecture for multimodal systems and provide a suite of general purpose, plug and play agents to handle essential tasks like speech I/O, fusion and semantic analysis and also including sub-components for multi-modal interaction for configurable distributed environment and to make the multimodal interface design more transparent, flexible and extensible. [6]

3. Related Work

Software design patterns are best practice solutions to common software problems. There are numerous eminent approaches for software architectures from design patterns and are in particular real time systems. These approaches just give unique descriptions of design patterns and high level guidance on how the patterns can be used to form software architectures. [3]

Distributed systems can be perceived as living organisms in the sense that the state of the computing system as well as its execution environment conditions change dynamically. In order to provide the intended services and functionalities at the required quality of service, adaptation of the system to the changing execution environment is necessary. Adaptive systems that can change their behaviour at run time have a number of potential benefits. For example, adaptive distributed systems can respond rapidly to improve the opportunities to optimize performance as the execution environment changes. [11]

To build distributed system & web applications the middleware methods & design methods are used. The middleware methods consist of communication, content & business process levels. At communication level, there are technologies that support tightly or loosely coupled communication styles. The various design methods are specifically proposed for concurrent & distributed systems. Due to their support for modularity, flexibility & reusability the object oriented methodologies have been proposed for the design of DSA. [3]

The experimental platform is demonstrate that is possible to offer multimodal interfaces in a distributed architecture responsible of the sensory input and output services, directed through a multimodal and interaction manager. Multimodal Architecture and Interfaces, improves some of its runtime framework sub-components and explores connecting the modality components to the framework as a remote service. [9]

4. Issues & Challenges

Different kinds of distributed systems work today, every pointed at tackling various types of issues. The challenges faced in building a distributed multimodal

system vary depending on the requirements of the system. In general, on the other hand, most systems will need to handle the following issues. [11] [12]

1. **Heterogeneity -** Different subsystem in the system must have the capacity to interoperate with each one in turn, despite distinctions in hardware architectures, operating systems (platforms), communication protocols, programming languages, software interfaces, security models and data formats. Due to multiple types of heterogeneity which poorly support for maintaining global consistency of data stores.

2. **Multimodality –** The multiple input/output channels and modes of accessibility it means combination of possibilities between data coming from different sources. Hence there is the complexity associated with building multi-modal systems.

3. **Transparency -** The whole framework should appear as a single unit and the complexity and interactions between the components should be typically hidden from the end user.

4. **Fault tolerance and failure management -** Failure of one or more components should not bring down the whole framework or entire system, and should be isolated. Replication of data and services provides fault tolerance and availability, but at a cost.

5. **Scalability -** The system may as well work effectively with expanding number of clients and expansion of a resource may as well enhance the performance of the system. Explicit and tedious programming related with data and network of applications on each device.

6. **Concurrency -** Shared access to resources should be made possible. So there is need of advanced software to realize the potential benefits.

7. **Openness and Extensibility -** Interfaces should be cleanly differentiated and openly accessible to empower simple extensions to existing components and include new components. But Poor middleware (Interfaces) support e.g. dis-connectivity, location independence, group collaboration, atomic transaction, quality of services (QoS).

8. **Migration and load balancing -** Permit the development of undertakings inside a framework without influencing the operation of clients or applications and distribute load among available resources for enhancing performance. But it affects on multiple network topology, bandwidth and Latency problem

9. **Security -** Security and privacy concerns regarding network communication. Access to resources should be secured to guarantee just known clients have the ability to perform permitted operations.

5. Conclusion

In today world practice, the popularity of distributed system applications is increasing day by day. As per discussion, there are many challenges in the distributed management system. So developers are focusing on the advancements of distributed system applications using various techniques. In distributed multimodal system, most of the interactions with computers take place using multiple I/O modalities. Input modalities are event-based or streaming based and it requires a user device to transfer human output into a form suitable for computer processing. To build a multimodal architecture based on a distributed architecture paradigm using third party services, we need an adaptive design pattern. Design patterns are one of such technology that is used in the design and development of such distributed multimodal architectural applications. In this paper we have presented some information about distributed multimodal system applications like the domains in which it is used and provide different types of services, the benefits of using it, the difficulties in designing the distributed multimodal system applications...etc. The main goal is to create a platform which is based on real time applications to interact with the users over a service-oriented architecture has to improve in some particular features. To provide multimodal interaction as well as tools allowing rapid creation of multimodal interfaces is then mentioned. Multimodal Architecture and Interfaces, improves some of its runtime framework sub-components and explores connecting the modality components to the framework as a remote service.

6. References

[1] Marc Pou, Luigi Ceccaroni, Barcelona Digital Centre Tecnològic Barcelona, Spain, "Multimodal Interaction in

Distributed and Ubiquitous Computing", IEEE (computer society 2010).

[2] Ronald Strebelow, Mirco Tribastone and Christian Prehofer, "Performance Modeling of Design Patterns for Distributed Computation", IEEE (computer society 2010).

[3] F.Dabous,F.Rabhi,H.Yu, "Using Software Architectures and Design Patterns For Developing Distributed applications", IEEE (ASWEC-2004)

[4] J. S. Fant, "Building Domain specific Software architecture from Software architectural design Patterns", ICSE-2011.

[5] C. Tianhuang, H. Feifei, Department of Computer Science and Technology Wuhan University of Technology, Wuhan, China, "Design Patterns Application in a distributed system", IEEE-2010.

[6] Sara Maalal, Malika Addou, Team of Systems Architecture, Laboratory of computing, Systems and Renewable Energy National and High School of Electricity and Mechanic ENSEM BP 8118, Oasis Casablanca, Maroc, "A new approach of designing Multi-Agent Systems with a practical sample", (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 2, No. 11, 2011.

[7] Yinsheng Li, Jianping Shen, Ying Huang, Weiming Shen, "Multi-Model Driven Collaborative Development Platform for Service-Oriented e-Business Systems", International Journal of Advanced Engineering Informatics (IJAEI), v. 22, no. 3, july2008, .pp 328-339.

[8] Duane Nickul, Laurel Reitman, James Ward, Jack Wilber, "Service Oriented Architecture (SOA) and Specialized Messaging Patterns-2007".

[9] Nicu Sebe, Journal of Ambient Intelligence and Smart Environments-1 (2009) 19–26, "Multimodal interfaces: Challenges and perspectives", IOS Press – 2009.

[10] Coulouirs, Dollimore, Kindberg, "Distributed system , concepts and design", 4th edition – Pearson, Addison Wesley publisher.

[11] FRED B. SCHNEIDER ON DISTRIBUTED COMPUTING, *IEEE DS Online*, Volume 1, Number 1 Interview by DejanMilojicic, dejan@spica.hpl.hp.com

[12] Krishna Nadiminti, Marcos Dias de Assunção, and Rajkumar Buyya, "Distributed Systems and Recent Innovations: Challenges and Benefits".

[13] Andy Crabtree, Terry Hemmings, and Tom Rodden, "Pattern-based Support for Interactive Design in Domestic Settings", ACM-2002.