A SURVEY OF SIMULATION TOOLS FOR PERVASIVE APPLICATIONS

R. Vivadha[1], J. Uma Maheswari [2],

[1, 2] *Assistant Professor, Department Of CSE, Rajiv Gandhi College of Engg & Tech., (India)*

ABSTRACT

Pervasive computing needs a poise of computing, design, and business constraints to be considered all through the design process. Realizing this creation involves a level of design in mixture of different fields that is not present in current pervasive design tools. Besides the progress of the key technologies for pervasive computing, the design of application itself has materialized as a remarkable research area. In spite of much growth, developing a pervasive computing application ruins a challenge because of in need of conceptual frameworks and supporting tools. In this paper, a survey has been conducted with different tools that support the integration of analysis and the design process which helps to identify design considerations. The investigation is done as of which tool involve both designers and engineers to take part in the design process. Evaluation is also performed for each tool with the conventional metrics of pervasive design tools. These evaluations afford imminent to key metrics and allow tool designers to recognize the requirements of their intentional spectators.

Keywords: Pervasive System, Context-Reorganization, Citycompiler, Diasuite, Sketchify.

I. INTRODUCTION

Pervasive Computing also known as ambient, physical, embedded, environmental or ubiquitous computing was first introduced by technology prophet Mark Weiser in the year 1991. He visualized a world of entirely coupled devices with economical wireless networks where information is available everywhere. A world in which computers and information technologies become invisible, and impossible to differentiate from everyday life: 'anytime, anywhere and always on' concept has been came into use. Today, a family is encircled by hundreds of 'invisible' devices in the machines around them. Likewise, in a pervasive computing environment, computers and information processing turn out to be normal, and pierce into every object in our day by day lives. Until lately the 'ubiquity' was rarely heard, but nowadays, it has promptly come to mean just about anything having to do with global connectivity. Consequently, information technology perspectives are no longer appreciated so much for the immersiveness they propose as for how tangential they emerge to be, and in this way dropping information overload. In 2004, McCullough renowned that architecture has obtained a digital layer, which occupies the design of organizations, services and communications and it looks as if that both architecture and interaction design mutually can help to compile the required structure for a improved assimilation. [5] The same as Edwards et al. depicted, the focus should be more on the 'value for end-users' than on the 'core technical workability'. This target conveys new challenges to the design and evaluation of pervasive applications

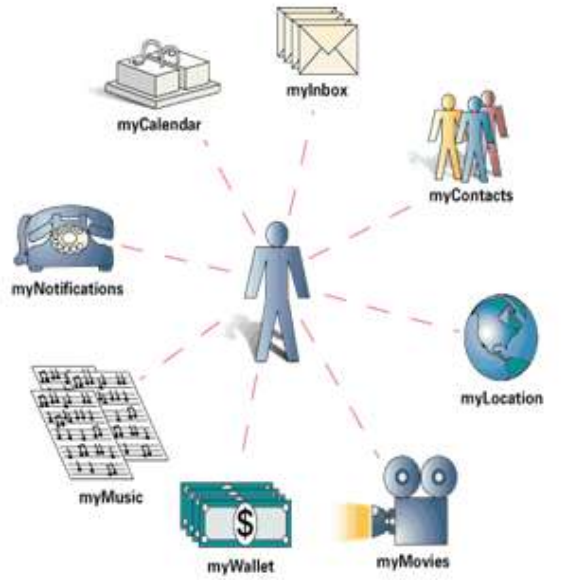

Fig- 1 Pervasive Computing

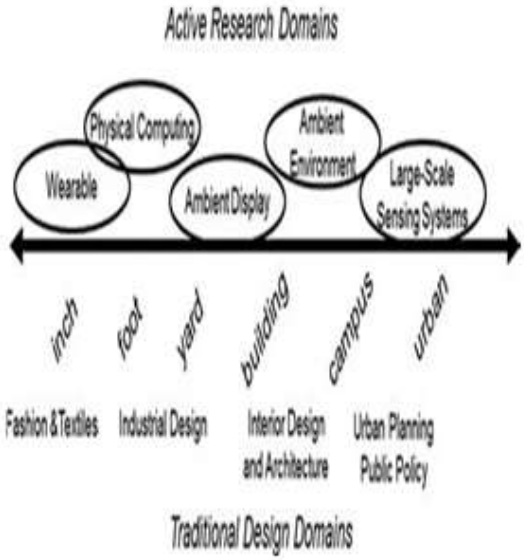

Fig 2: Taxonomy of Pervasive Computing Domains

As a design needs to determine both the technical and users' prompt features to maintain the acceptable user skills, the developers have to build the overall scenarios, so as to recognize the day by day practices of their users. Conversely, the day by day conditions are constantly varying, owing to the diverse perspective of use and the settings for communication, which causes the challenging prophecy in producing a set-up. For example it is extremely tricky to envisage how users will respond when designing an interactive service. Presently, the technical features for constructing pervasive applications are noticeable, due to the existing sensing, data processing and context-recognizing technologies. However for researchers, there is still in need for the criteria to design and estimate the features of the application itself. The purpose of features in a design today mainly depends upon the designers' practices and perceptions or on the particular styles supported by the particular infrastructure systems. It can also be evaluated that a good quality design of pervasive applications can quickly fix on to which elements should be limited within a given application and assess how well those features will put in value for users to tackle their requirements. It aids the designers in speeding up an iterative development process and in accepting the full intricacy of applications by setting the exact reasons in each design phase. In such cases, constructing a prototype is a core means, which permits the designers to reveal, assess or test a developing design with the negligible effort. [8]

To appropriately implement pervasive systems, the designers must design them to tackle human desires and concerns. Design needs a considerate and harmonizing of appropriate constraints to find a suitable solution. Adding up to this, the vast scale of pervasive computing means this design will occur across multiple domains together with fashion, industrial design, architecture, urban planning. [7]

Today, numerous tools for pervasive applications are present to satisfy the miscellaneous necessities devolved into the entire design process: from sketching the idea early with a low-fidelity prototype to deploying a high fidelity prototype and testing it in a practical environment. Compared to those more established application domains in pervasive computing, e.g. the middleware for prevailing over the heterogeneity via uniform interfaces, design tools are still at an early stage. [8]

II. KEY REQUIREMENTS FOR PERVASIVE COMPUTING APPLICATIONS

Moreover to involve proficiency on underlying technologies, mounting a pervasive computing application also entails domain-specific architectural knowledge to gather information pertinent for the application, develop it, and execute actions. Some of the key requirements for developing pervasive computing applications are reviewed below: [3]

2.1 Abstracting Over Heterogeneity

Pervasive computing applications interrelate with entities whose heterogeneity has a tendency to penetrate in the application code, messing it with low-level details. [3] This condition need to raise the level of abstraction at which entities are raised, to factor entity variations out of the application code, and to protect it from distributed systems dependencies and communication protocol details.

2.2 Architecturing an Application

Theoretically, pervasive computing applications gather context information, process it, and carry out actions. Software development methodologies such as model driven engineering are also useful to design pervasive computing applications. [3] A prominent example is PervML which relies on the general-purpose modeling notations of UML to produce particular programming support. However, such approaches do not afford a conceptual framework to direct the design.

2.3 Leveraging Area-Specific Knowledge

Since the pervasive computing domain comprises an increasing number of areas, information about each area wants to be shared and made reusable to make easy the growth of applications. Reusability is required at two levels. [3] First, it is looked-for the entity level because applications in a given area frequently share the same classes of entities. Second, reusability is looked-for the application level to facilitate the developer to act in response to new requirements by using obtainable context computations.

2.4 Covering the Application Development Life-Cycle

Existing general-purpose design frameworks are standard and do not wholly sustain the development life-cycle of pervasive computing applications. To cover this life-cycle, a design framework explicit to the pervasive computing domain is required. This domain-specific design framework would get better yield and assist progress. [3] To make this design framework effectual, the conformance between the requirement and the implementation must be definite. After the application is implemented, tools should aid for all characteristic of its consumption. Maintenance and evolution are important matter for any software system. They are even more significant in the pervasive computing domain where new entities may be organized or detached at any time and where users may have varying needs. These maintenance and evolution phases should be supported by pervasive tools.

2.5 Simulation of the Environment

The use of a pervasive computing application involves abundant equipments to be obtained, tested, configured, and installed. In addition, some scenarios are hard to test because of the situations involved. [3] To overcome this operation barrier, tools should be made available to the developer to test pervasive computing applications in a computer-generated environment.

III. PROPERTIES OF PERVASIVE TOOLS

An overview is made by categorizing the pervasive design tools by its properties which are discussed below: [7]

3.1 Multiple Representations

A key obscurity in design is the language or representation used to articulate a design. [3] Often interactive products are produced using a textual programming language, which is well-known to the engineers, but unknown to the designers. Having multiple, and probably concurrent, representations of a design defeats this problems by letting all members of a design team to take part in the design of an application and have a view that is appropriate for them.

3.2 User Defined Events

Pervasive applications depend greatly on sensor data collected from the physical world. These physical events can be as easy as the state of a switch, or they can be more difficult events such as the classification of an activity. [7] Apart from the event in difficulty, blending sensor data into significant information is a tricky task. For easy events, thresholds can be set on sensor data, but more difficult events must be examined with a machine learning algorithms. Several tools have developed methods to assist in fusing events from complex sensor data.

3.3 Knowledge Support

Including sensing and computing brings in a new material into the design process that is unknown to non-computing authority.[7] Dow et al. prompts this category by conversing how specialized designers necessitate information about current and imminent technologies to notify their design. Furthermore, when performing collaborative design, all parties must recognize how the basic technology behaves.

3.3 Testing and Device Support

Depending on the size of a pervasive system it may not be probable to have all devices or services there during design. [7] As well, the system will have to be tested using real or generated actions.

3.4 Integration with Current Practice

At last, tools are examined that challenge to integrate current design practice into the design of pervasive applications.[7] This guideline articulates to the heart of design and tries to use existing design practices to produce interactive computing elements. This approach continues designers within the normal dominion of design, permitting comfort and quick iteration.

IV. SURVEY OF DESIGN TOOLS

In this section we present a survey of design tools for pervasive computing. [7] In Section III we summarized the properties by which to evaluate the design tools.

4.1 Diasuite

DiaSuite is a tool suite which covers the development life cycle of a pervasive computing system. This tool suite consists of a domain-specific design language, a compiler for the language that generates a Java programming framework, an editor to identify simulation scenarios, and a 2D-renderer to imitate pervasive computing

applications. This tool uses a software design approach to make the development process. DiaSuite presents a language, DiaSpec, devoted to architecturing pervasive computing systems. [1] DiaSpec lets an area authority to define categorization by asserting the necessary entities of the system. DiaSpec also offers to declare the architecture of the system in the form of context and controller components. A simulation editor and 2D-renderer are also part of DiaSuite to imitate the resulting pervasive computing application.

4.2 Sketchify

Sketchify is a tool for sketching user interfaces. It gives designers the liberty to operate interactive materials by uniting elements of traditional freehand sketching with functional extension and end-user programming tools, such as spreadsheets and scripting. [6] This tool has a number of features potentially helpful for sketching of interactive systems, as well as support for looking at complex technologies in an easy way, multiplicity and reuse of existing environments. Some of the benefits of using Sketchify tools are investigate the possibilities and limitations of technologies, mixture of components, extensibility and domain independence, reuse of existing environments, diversity of development styles and avoiding proprietary lock-in, promoting more efficient collaboration between designers and engineers.

4.3 Modkit

Modkit is a tool that makes it possible for learners and knowledgeable programmers /designers to fetch tangibles to life by contributing graphical command blocks stimulated by the Scratch programming environment. Modkit as a means for designing systems that entails sensing, actuation, programming, embedded design, and crafting. Participants will look for Modkit programming by using Crimp Cards easy-to-assemble kits of hardware components. [3] It has as a feature found and manufactured materials into interactive projects by combining them in unpredicted ways. It interprets opinion from different types of sensors into light, motion, or sound. It expands Modkit activities for future designers from all backgrounds. It expands Modkit to improved support certain applications or user groups.

4.4 Inspirational Bits

Inspirational bits as a way to turn out to be more well-known with the design material in HCI, the digital material. The inspirational bits are described as rapid and filthy but fully working systems in both hardware and software constructed with the aim of revealing one or several of the dynamic properties of a digital material. It can also be used as one of the preliminary steps in a design process, making them alike to a technology-driven design process. [10] These inspirational bits should be rapid to build. Whereas building the first bit in a material may take longer time, nearly all of the digital materials are very flexible and from our experience the second and third bit will take much fewer times to build. By this, we also believe the overall amount of time it takes to build interactive systems in fact will be shorter, in that we will keep away from fighting our material and instead functioning with it functioning out the design concept.

4.5 City Com Piler

CityCompiler is an integrated environment for the iteration-based development of spatial interactive systems. [9] It envisages interactive systems in a virtual 3D space by combining the Processing source code and the 3D model of the real space, designed with Google SketchUp. Hybrid prototyping would be useful not only for organize a system into the real world but also for designing a system with a new concept.

Tool name	Multiple Representations			User defined events		Knowledge support	Testing and device support	Integration with current practice	
	Visual and textual representations	States and code	Tangible	Demonstration	Infrastructure/API		Emulation and simulation	Sketching interaction	Field observations and storyboard
Diasuite	✓	✓	✓	✓	✓		✓		
Sketchify	✓	✓	✓					✓	✓
Modkit	✓	✓	✓						
Inspirational bits						✓			
CityCompiler	✓	✓		✓	✓				✓
d.note	✓	✓	✓	✓	✓		✓		
I*Catch	✓	✓	✓						
ESPranto SDK	✓	✓	✓		✓				
Xtel				✓	✓				
Makeit				✓	✓		✓		

Table 1: Comparison of Different Simulation Tools

4.6 d.note

d.note is a revision tool for user interfaces articulated as control flow diagrams. d.note initiates a command set for altering and interpreting both look and performance of user interfaces; it also describes execution semantics so proposed changes can be tested instantly. With d.note, users can set up alternatives for manifestation and application logic. [2] d.note signifies the alternatives by duplicating the original state and visually summarizing both original and alternative. d.note was implemented as an extension to d.tools.

4.7 i*CATch

i*CATch wearable computing framework, was developed particularly for children and beginners to the field.[6] The i*CATch framework is based upon a bus-based architecture, and is further scalable than the existing alternatives. It comprises of a set of plug-and-play components, a construction platform with a homogeneous interface, and an easy-to-use hybrid text-graphical integrated development environment. The i*CATch construction kit was intended for the two reasons of sustaining creativity and make possible standardization in wearable computing.

4.8 ESPranto SDK

The ESPranto SDK is a part of the Edutainment Sensor Platform (ESP) which supports the growth of sensor/actuator based applications, most particularly in the domains of educational toys, games and lighting.[11] ESPranto SDK is a toolkit that provides beginner programmers to build up simple applications, non-technical domain experts to build up professional content, and technical experts to extend complex building blocks for the other users to include in their applications. The SDK supports the end user's progression from trainee to expert programmer. The preclusion of runtime errors is a patent benefit of the SDK. To trim down development time and costs, the SDK must allow all users to develop applications with no or slight help from a software engineer. The SDK must permit users to switch effortlessly to a higher level of programming, i.e. without having to adjust their attitude. The SDK must be adaptable, in that it is not essential to produce a new tool for each hardware grouping.

4.9 XTel

XTel encompasses three tools: Moxa, Talktic and Entity Collaborator. The 'moxa' Micro Control Unit (MCU) board connects to sensors and actuators and is competent of short-distance wireless communications; the 'Talktic' programming/runtime environment for the MCU board that holds a JavaScript parser, compiler, VM and library; and the 'Entity Collaborator' P2P network library that is capable of managing continuous information such as video and audio in addition to the discrete information from sensors.[12] The utilization of these tools allows both developers and designers to rapidly and effortlessly generate ubiquitous contents. 'XTel' is a progress support environment that facilitates the proficient making of these ubiquitous contents. The main drawback is that the consumers who are not familiar with programming cannot use it, because it is not easy to build up visual expression such as Processing.

4.10 MAKEIT

The MAKEIT framework an acronym for 'Mobile Applications Kit Embedding Interaction Times' is used to make functional, hi-fi prototypes for mobile devices following sophisticated interaction techniques. It simply produce and modify applications while at the same time provides support in keeping proposed end user interaction times low. [9] An integrated development environment is used for hi-fi prototyping of mobile phone applications for producing a code framework for the ultimate implementation. A primary model based on state graphs confirms parts of the application logic and perceives faults in the navigational structure and propose alternatives. An integrated model is estimated for task completion times early on the design without calling for organizing a prototype on the actual target hardware platform.

V. RELATED WORK

The design of tools for pervasive computing mainly focuses on the ability and forms. Another consideration for the design of tools is based on how these tools are active in the design process life cycle. The design process life cycle is structured, linear cyclic and iterative. The key property of design tool is its flexibility. This paper mainly focuses on the properties needed for a tool that have various perspectives such as multiple representations, knowledge support etc., Tools that are used for designing must have the attribute of knowledge support so that the user can understand sensing technologies without implementation. The tools must allow for the expression of different ideas and also must support for different domains.

ESPranto SDK attempts to introduce beginners to program. However, domain specific tools such as iCATCH deals with wearable's and location based applications. When using multiple representations, the issues are the amount of linkage or coupling between the two representations. Tools such as DiaSuite, Sketchify, ESPranto SDK, Modkit supports visual and textual representations, state and code, Tangibles. These tools provides graphical user interface and make the user to generate the code efficiently. DiaSuite and Modkit are used as simulating tools. Simulation allows for a richer design experience that can test features such as user interaction and network performance estimation.

VI. CHALLENGES

6.1 Design Representation

While one of the attributes in the multiple representation of a single design, the issues in the determination of which representation is to use. The two parts in the multiple representations are visual and textual representation. Comparing these two representations, textual programming provides explicit control and conditional events which are more difficult to describe in a visual language where as visual representation is easier for beginner programmers and can more easily express continuous behaviors.

6.2 Simulation Environment:

A simulator for pervasive environment needs to serve three roles in the design process. They are as follows: (1) simulating the input space of an application, including the explicit (e.g., mouse or keyboard events) and implicit input (e.g., location sensed input when user moves); (2) simulating the logical control flow that jumps between sensors, servers, handhelds (such as PDA) and any other kinds of networking appliances; and (3) simulating the output space of an application, which means to visualize the environment effects caused by the application behaviors.

6.3 Understanding Context Awareness

Designers who are already familiar with the language can now build context-aware applications. However, the drawback is that the delivered context is not the desired information or is not suited for a particular application. In such case, the context recognition algorithms will need to be re-evaluated, which may cause major difficulty. Recent work has been conducted to provide end-users intelligibility in context aware systems (Lim and Dey, 2010) such that they can realize why certain actions were and were not taken. A similar intelligibility approach is required to allow design teams to understand the behavior of recognition algorithms, and allow the parameters of the algorithm to be exposed and modified.

VII. CONCLUSION

In this paper, we have done a survey of the current studies in designing and evaluating pervasive applications. We have listed some of the key requirements to design a pervasive computing application and also some properties of pervasive tools. We have also summarized the need for design of pervasive computing systems and performed a survey of some tools.

REFERENCE

[1] Benjamin Bertrana, Julien Bruneaua, Damien Cassoua, Nicolas Loriantb, Emilie Ballanda, Charles Consela, "DiaSuite: a Tool Suite To Develop Sense/Compute/Control Applications", May 2012.

[2] Björn Hartmann, Sean Follmer, Antonio Ricciardi, Timothy Cardenas, Scott R. Klemmer, "d.note: Revising User Interfaces Through Change Tracking, Annotations, and Alternatives", ACM Transactions, April 2010.

[3] Damien Cassou, Julien Bruneau, Charles Consel, Emilie Balland, "Towards A Tool-based Development Methodology for Pervasive Computing Applications", March 2012.

[4] Edward Baafi, Amon Millner, "A Toolkit for Tinkering with Tangibles & Connecting Communities", ACM Transactions, Jan 2011.

[5] A. Fatah gen. Schieck, A. Penn, V. Kostakos, E. O'Neill1, T. Kindberg, D. Stanton Fraser, T. Jones, "Design Tools for Pervasive Computing in Urban Environments", *DDSS 2006.*

[6] Grace Ngai, Stephen C.F. Chan, Vincent T.Y. Ng, Joey C.Y. Cheung, Sam S.S. Choy, Winnie W.Y. Lau and Jason T.P. Tse, "i*CATch: A Scalable, Plug-n-Play Wearable Computing Framework for Novices and Children", ACM Transactions, April 2010.

[7] Jason B. Forsyth, Thomas L. Martin, "Tools for interdisciplinary design of pervasive computing", International Journal of Pervasive Computing and Communications, Vol. 8 Iss: 2 pp. 112 – 132.

[8] Lei Tang, Zhiwen Yu, Xingshe Zhou, Hanbo Wang, Christian Becker, "Supporting rapid design and evaluation of pervasive applications: challenges and solutions", Springer 2011.

[9] Paul Holleis and Albrecht Schmidt, "MAKEIT: Integrate User Interaction Times in the Design Process of Mobile Applications", Springer 2008, LNCS 5013, pp. 56–74.

[10] Petra Sundström, Alex S. Taylor, Katja Grufberg, Niklas Wirström, Jordi Solsona Belenguer, Marcus Lundén, "Inspirational Bits Towards a Shared Understanding of the Digital Material", ACM Transactions, May 2011.

[11] Robert van Herk and Janneke Verhaegh, Willem Fontijn, "ESPranto SDK: an Adaptive Programming Environment for Tangible Applications", ACM Transactions, April 2009.

[12] Satoru Tokuhisa, Takaaki Ishizawa, Yoshimasa Niwa, Kenji Kasuya, Atsuro Ueki, Sho Hashimoto, Kazuhiko Koriyama and Masa Inakage, "xtel: A Development Environment to Support Rapid Prototyping of "Ubiquitous Content"", 2009.

[13] Yasuto Nakanishi, Koji Sekiguchi, Takuro Ohmori, Soh kitahara, and Daisuke Akatsuka, "Hybrid prototyping by using virtual and miniature simulation for designing spatial interactive information systems", 2011.

[14] Zeljko Obrenovic and Jean-Bernard Martens, "Sketching Interactive Systems with Sketchify", ACM Transactions on Computer-Human Interaction, Vol. 18, No. 1, Article 4, Publication date: April 2011.