ECONSPEAK: A Journal of Advances in Management IT & Social Sciences,
Vol. 7, Issue 4, April 2017 Impact Factor 5.132 ISSN: (2231-4571)
www.skirec.org Email Id: skirec.org@gmail.com

Need and Challenges for Software Engineering in Pervasive Computing

Gurmeet Kaur, Assistant Professor,
Dyal Singh College, Karnal

Abstract-Moving away from decades of machine-centric computing and designing pervasive human-centric computing, the new wave of computing, a reality revolutionizes the relationship between humans and computing systems. The growing interest in the use of context-awareness as a technique for developing pervasive computing applications that are flexible, adaptable, and capable of acting autonomously on behalf of users. In order to implement the vision of pervasive human-centric computing, it is necessary to reform software engineering education to well prepare graduates of software engineering programmes for the new opportunities and challenges of software engineering in the pervasive computing era. The software challenges to turn such pervasive or ubiquitous computing environments into reality are enormous, to say nothing of software , hardware and social challenges. In this paper, we review some of the work of software components and analyze where our solutions are lacking and must be adapted for pervasive computing..

1. Introduction

It is widely acknowledged that pervasive computing introduces a radically new set of design challenges as compared with traditional desktop computing. In particular, pervasive computing demands applications that are capable of operating in highly dynamic environments and of placing minimal demands on user attention. late Michael Dertouzos, Director of the MIT Laboratory (1974-2001) for Computer Science, who pioneered MIT Project Oxygen to make pervasive human-centric computing a reality, pointed out the need for pervasive computing. He stated the following:

If computers are to live up to the promise of serving us, they will have to change drastically and never again subject us to the infuriate experiences we all have shared [1].

It is envisioned that pervasive computing systems will help people to achieve more while doing less. These systems will:

- Understand user when he speak to them;
- Do much of our routine brainwork for user;
- Get us the information for user when and where we want it;
- Help us work with other people across space and time;
- Adapt on their own to our individual needs and desires [1].

In the pervasive computing era, there will not need to carry our own physical devices with us any more. Instead, configurable generic devices, either embedded or handheld in the environment, will bring computation to user, whenever he need it and wherever he might be. As user interact with these *anonymous* devices, they will adopt our information personalities. They will respect our desires for security and privacy. The user need not have to type click, or learn new computer jargon. Instead, he will communicate naturally, using gestures and speech that describe user intent (*send this to Soni* or *print that picture on the nearest colour printer*), and leave it to the computer to carry out our will [2].

Pervasive human-centric computing systems will change how businesses, organizations and governments work with each other, as well as how individuals interact. It represents the dawn of a new era in Information Technology (IT) [1].

To shift the focus of computing from machines to humans, major changes are required not only in technologies and systems, but also in the approach to deploying , developing and managing technologies and systems. Weiser presented his vision for pervasive human centric computing in 1991. He further articulated his vision as follows:

There is more information available at our finger tips during a walk in the woods than in any computer system, yet people find a walk among trees relaxing and computers frustrating. Machines that fit the human domain instead of forcing humans to enter theirs will make using a computer asrefreshing as taking a walk in the woods

[4][3].

2. Motivation

Challenges which are based on natural characteristics of pervasive computing systems (i.e. dynamism , mobility, and heterogeneity) can be evaluated from a more domain specific perspective, that is, e-learning in our case. E learning refers learning which uses muliple technologies such as internet, television etc. in a manner pointed out by [5]:e-*enhancements of models of learning. That is to say that; using technology to achieve better learning outcomes, or a more effective assessment of these outcomes, or a more cost-efficient way of bringing learning environment to the learners* [5].

Hence, we particularly list following basic interrelated requirements for such pervasive learning environments:

(1) *device independence*: applications and data should be always accessible without any dependence on device

(2)*application independence*: data should be always accessible without any application dependence,

(3) *adaptivity and adaptability*: learning environment and elements of this environment should dynamically adapt according to context of learner(s) and users should be able to configure such environments such as composing/decomposing applications and data

(4) *collective operation*: applications in such domain must be able to collectively operate for the benefit of users in a seamless manner. Adaptivity is long studied both in adaptive web systems and adaptive e-learning systems [6], and in such systems adaptivity is generally considered as an aspect between user and application based on user profiles and models.

Pervasive human-centric computing systems are dedicated systems that are capable of sensing, measuring, monitoring, predicting and reacting to physical world conditions. To support a wide range of human activities, pervasive human-centric computing systems must be:

Pervasive: should be available everywhere and accessing the same information base through every portal .

Nomadic: allowing users and computations to move around freely to meet the users' needs;

Embedded: sensing and affecting the physical world;

Adaptable: providing flexibility and spontaneity in response to changes in the operating conditions and user's requirements ;

Intentional: enabling people to name services and software objects by intent;

Powerful yet efficient: freeing itself from restriction imposed by bounded hardware resources, addressing system constraints imposed by user demands and available power or communication bandwidth;

Eternal: never requiring shut down or reboot while components are added or removed in response to errors ,demands, or upgrades [2].

In a pervasive computing environment, user and perceptual technologies will directly address human needs and consist of the following:

Knowledge access technologies: offering vastly improved access to information and customized to the needs of users (ie people, applications and software systems);

Collaboration technologies: enabling the formation of spontaneous collaborative regions that accommodate the needs of mobile people and computations, and also provide support for recording and archiving video and speech fragments from a variety of sources and/or events;

Perceptual technologies like speech and vision technologies: enabling communication with devices, networks and software to extend the range of user technologies delivered to all places.

Automation technologies: offering natural, easyto-use, customisable and adaptive mechanisms for automating and tuning repetitive information and control tasks;

3.Context Aware Pervsive computing

Context aware computing aims to enable device to provide better service for people through applying available context information [7].

Above a generic definition of context aware computing is given, which emphasizes the relation between user, context and computing, but how do one apply available context information? Although multiple categorizations for context-aware systems are already given [8], one can prefer to re-interpret these categorizations based on adaptive systems, particularly according to adaptive web systems. This is because one can defined adaptivity as a key factor of intelligence and as a key relation between context and computing for context-aware computing systems. Therefore by referring to [8] and the field of adaptive web [6] for categorization of context aware computing applications, we propose below categorization:

(1) *context based filtering and recommendation of services and information*: examples might include accessing the history of a nearby object , finding the nearest printer etc.,

(2) *context based service and information searching*: e.g. location aware query rewriting for a search for available restaurants (query rewriting is a technique used in adaptive web systems for information filtering by rewriting a user query according to the user profiles) etc.,

(3) *context based presentation and access of information and services*: e.g. selecting voice when screen displays are not available (multimodal information presentation and user interfaces), dynamic user interfaces etc.,

(4) *context adaptive navigation and task sequencing*: adaptive navigation is a technique employed in adaptive web systems. The user can extend this idea in pervasive computing since a user's interaction might consist of multiple related sub-tasks in relation with his goals and might lead to context aware task sequencing,

(5) *context based application and services modification/configuration* : this need mainly arises from varity of devices available in the environment, e.g. disabling particular features depending on the capabilities of target device,

(6) *context based resource allocation*:
this might include allocating physical recourses (e.g. memory, even non-hardware physical resources) for the use of other entities in the setting (e.g. , users , applications etc.).

(7) *context based actions*: [9] proposes three levels of context dependent automatic actions: manual, semi-automatic, [10] and notes that fully automatic actions based on context are rarely useful, and incorrect actions can be frustrating.

It is worth to note that, adaptive behaviors of context aware systems are not necessarily need to depend on the present context, rather such systems should also be able to adapt dedicately by making use of present context or historical context to predict future context of the setting. An example is given in [11] where a user walks through the building and submits a printing request, the selected printer should not depend on the user's current location but rather to his final destination. According to presented categorizations and elaborations, we extend previous definition of context-aware systems as follows:

Context aware computing aims to enable better service delivery through proactively adapting access,use, structure and behavior of information, applications, services and physical resources with respect to available context information.

An up-to-date and specific example is a famous social networking website, Facebook. This web application provides users with the contextual information of their network (by means of notifications) like who watches, reads what or who becomes friend with whom. In this way users can identify people with similar likes and arrange their own environment accordingly. Such case is also of use in the domain of e-learning, a system can provide users with the contextual information of the environment and other learners like who read what, who knows what, who takes the same courses or who works on the same problem, so learners can find appropriate mentors or construct a learning path for themselves. Such approach might be called as "*environment awareness"* for users which is counterpart of context-awareness for machines.

4. Strategies for Software Engineering challenge in pervasive computing

As we know that to achieve real life application of pervasive computing is challenging task. To implement this a lot of challenges have to resolved for software engg. discipline and software component. The suggested core strategies for software engineering education reform include the following:

1. Redesign of software engineering curricula by integrating pervasive human-centric computing and autonomic computing into the curricula;
2. Systematic integration of applied and experimental research in software engineering for pervasive human-centric computing into software engineering education;
3. Industry-academic partnerships in both research and education;
4. Engaging students in cross-disciplinary research and development;
5. Institutional support and funding for cross disciplinary collaborations in research and education;
6. Fostering life-long learning;
7. Systematic updating of the contents and structure of software engineering curricula.

It is necessary to restructure software engineering curricula by integrating pervasive human-centric computing and autonomic computing into the curricula [12][13][14].

The rapidly evolving and multidisciplinary nature of pervasive human-centric computing and autonomic computing requires the systematic integration of applied

and experimental research into software engineering education to enhance students' learning experiences. Engaging software engineering students in applied and experimental research helps them to acquire invaluable experience that they cannot gain by simply reading technical articles and attending lectures.

To further enhance students' learning experiences, it is crucial to develop and nurture industry-university partnerships in research and education. This will also help students to work with industry sponsors while enhancing their hands-on experiences, as well as their technical competences and skills [12][13][14-18].

Furthermore, the multidisciplinary nature of pervasive computing requires collaborations in educational and research activities among field experts from different areas, as explained in earlier parts of this article. Hence, it is essential for engineering educational institutions to foster cross disciplinary collaborations in research and education so that students can engage in collaborative, multidisciplinary projects with faculty and other field experts and professionals across various fields from universities, industry and research organizations. This will also help students to enhance and learn their engineering knowledge and skills, as well as their professional skills (e.g. teamwork, written and verbal communications, etc).

Collaborative multidisciplinary projects require extra efforts to ensure effective and productive cooperation among all the people involved. Thus, it is critically important to change the culture, funding
structure and faculty performance evaluation system
in academia to provide the necessary institutional support and funding for cross-disciplinary collaborations among faculty from different departments, colleges, and universities, and other researchers from industry and non-academic institutions.

Due to the nature of software engineering for pervasive computing and autonomic computing, software engineers need to be strongly committed to life-long learning and regularly update their technical knowledge, competences and skills. To help graduates become self-motivated and life-long learners, it is crucial to provide students with opportunities to acquire both the awareness of the necessity of life-long learning and the knowledge, skills and abilities to engage in life-long learning.In order to ensure that software engineering educational programmers provide the best learning opportunities for students, it is crucial to maintain the flexibility of software engineering curricula, and to update systematically the contents and structure of the curricula.

4.1 Software components for pervasive computing

The pervasive computing environment drive us to face the need for components and their boundaries more clearly. Pervasive services will have to be composed from individual "components" residing in the large number of heterogeneous computing elements. The hardware domain itself will drive a natural boundary between components. This may be the most clear-cut definition of a component. A component will be an independently deployable piece of software that resides on one hardware component and provides a service element heterogeneity

The most striking characteristics of software components in the pervasive computing environment are the need to deal with heterogeneity and the need for dynamic (ad hoc) adaptation to, and interaction with, communicating components. Current component models are homogeneous in the kind of components for example, JavaBeans components are for desktop environments while Enterprise JavaBeans are for server and enterprise-wide components. To make application evelopment manageable, we probably need a single component model that is "scalable" in the sense that it supports the development of components of various granularity, components that can reside in tiny computing elements.

[Jazayeri95]. While language-specific components are still important, the pervasive environment requires us to also deal with heterogeneous components. The work of Johann Oberleitner [Oberleitner01] deals with the heterogeneity of component models. He has designed and built the Vienna Component Framework (VCF) that captures the essential characteristics of different component models such as simple X-Windows components ,COM, CORBA, JavaBeans, Enterprise JavaBeans. The VCF provides foundational support for (CBSE) component-based software engineering. It is used as the lower layer of a CBSE environment called the Component Workbench [Oberleitner02]. The Component Workbench provides transparent access to each component model and allows applications to be built from components coming from different component models. It also supports the user in maintenance activities such as replacing components with other components. For example, if an application is to be moved to an environment where a needed

component is not available, that component can be replaced with an equivalent component in the new environment. Consider an application that is built in the MS Windows environment and it uses the Internet Explorer as a component. In moving the application to a UNIX environment, the browser can be replaced by a Mozilla or Opera browser. The Component Workbench supports this replacement by helping the user match interfaces, methods, and attributes from one component to another component.

The Vienna Component Framework captures the common characteristics of different component models by providing a uniform type system across component models . It supports an event-based communication mechanism for the interconnection of components. It also offers an architectural description language that describes the composition of an application in terms of components.

One of the key problems of building applications out of components—component based software engineering—is what to do if the component you need is not available in the catalogs you have. Clearly, no catalog will have every component that an application developer needs. But, often, there will be a related component, or one that is "almost" the one needed. There are several possible paths to take in this case. One is for the developer to modify the related component to make it fit user needs. This approach defeats the purpose of component-based development because of the fundamental reason that it breaks the separation of concerns between component development and component usage. A more effective approach is to automatically "adapt" the existing component to the need of the application. Ideally, with automatic adaptation, the component developer can provide a minimal catalog of components but the user gains the benefits of a larger catalog. The goal in the component work reported in [Jazayeri95] was to use generic programming to build powerful yet minimal catalogs. Thomas Gschwind's dissertation [Gschwind02] concerns the topic of automatic component adaptation and introduces a particular kind of adaptation called "type-based adaptation."

Modern languages such as Java, and modern component models such as Cobra Components support strongly-typed components and provide mechanisms for querying the type of the component at run-time and type-based adaptation exploits these features to automate the adaptation process. The Component Workbench uses type-based adaptation to support the replacement of components from one component by components from another model.

Type-based adaptation is also a good fit for pervasive computing environments because the communication protocols they use and the devices that need to communicate with each other are not known a –priori.

The Vienna Component Framework and type-based adaptation are clearly two important ingredients for dealing with the heterogeneity of components that will have to face. But they are only preliminary steps towards meeting the wide heterogeneity and dynamicity that is expect to face pervasive computing environments. Components will have to compose dynamically and adapt dynamically. Versioning and legacy issues associated with such dynamically evolving services and their components will pose enormous challenges for software engineers.

5. Conclusion

The vision for pervasive human centric computing, the new wave of computing, cannot be implemented without software engineering programmers. This article presents the necessity of integrating pervasive computing into software engineering curricula and presents a set of suggested core strategies for integrating pervasive into software engineering education. The suggested core strategies include redesigning software engineering curricula

To incorporate pervasive computing into the circular systematically integrating pervasive computing the search into education, engaging students in applied and experimental research ,establishing and nurturing industry-academic partnerships in research and education, providing institutional support and funding for cross-disciplinary collaborations in the search and education, systematically updating the contents and structure of software engineering curricula to better prepare students for the new challenges and opportunities of software engineering and software component to make feasibility of pervasive computing in real life application era.

6.REFERENCES

[1] Dertouzos, M.L., *The Unfinished Revolution: Human-Centered Computers and What They Can Do for Us*. New York: HarperCollins (2001).

[2]. MIT Project Oxygen, http://www.oxygen.lcs.mit.edu/

[3]. Weiser, M., The computer for the twenty-first

century. *Scientific American*, 265, **3**, 94-104 1991

[4]. Weiser, M., Some computer science issues in ubiquitous computing. *Comm. ACM*, 36, **7**, 75-84 (1993).

[5] T. Mayes, S. de Freitas, "Review of E-Learning Theories, Frameworks and Models," *JISC E-Learning Models Study Report*, Joint Information Systems Committee, www.jisc.ac.uk/elp_outcomes.html, 2004.

[6] P. Brusilovsky, A. Kobsa, W. Nejdl (Eds.), *The Adaptive Web*, Lecture Notes in Computer Science, Springer-Verlag, Berlin, Germany, 2007.

[7] L. Han, S. Jyri, J. Ma, K. Yu, "Research on Context-aware Mobile Computing", *Proceedings of Advanced Information Networking Applications Workshops*, pp 24-30, 2008.

[8] J. Pascoe, "Adding generic contextual capabilities to wearable computers", *Proceedings of 2nd International Symposium on Wearable Computers*, pp. 92-99, 1998.

[9] J. Mäntyjärvi, U. Tuomela, I. Kansala, J. Hakkila, "Context-studio–tool for personalizing context-aware applications in mobile terminals", *Proceedings of Australasian Computer Human Interaction Conference*, Addison-Wesley Longman, pp. 64-73, 2003.

[10] P. Korpipää, J. Hakkila, J. Kela, S. Ronkainen, I. Kansala, "Utilizing context ontology in mobile device application personalization", *Proceedings of Mobile and Ubiquitous Multimedia*, ACM Press, pp. 133-140, 2004.

[11] J. Coutaz, J. Crowley, S. Dobson, D. Garlan, "Context is Key", *Communications of the ACM*, 48(3), March 2005.

[12] Pour, G., Pervasive computing reforming software engineering education. *World Trans. on Engng. and Technology Educ.*, 2, **3,** 357-360 (2003).

[13]. Pour, G., Restructuring software engineering education towards cross-disciplinary collaborations to create pervasive information systems and technologies. *World Trans. on Engng. and Technology Educ.*, 3, **1**, 85-88 (2004).

[14]. Pour, G., Expanding the horizons of software component Workbench," Proceedings of the 3rd Intl. workshop on Software Engineering in Middleware , Jan 2002.

[Gschwind02] T. Gschwind, *Adaptation and Composition Techniques for Component-Based Software Engineering*, PhD dissertation, Distributed Systems Group, Technical University of Vienna, 2002.

engineering education: integrating autonomic computing into the curriculum. *World Trans. on Engng. and Technology Educ.*, 5, **1**, 179-182 (2006).

[15]. Pour, G., Griss, M. and Lutz, M., The push to make software engineering respectable. *IEEE Computer*, 33, **5**, 35-43 (2001).

[16]. Pour, G., Engineering curriculum reform to introduce students to security and privacy in the Internet era. *World Trans. on Engng. and Technology Educ.*, 4, **2**, 285-288 (2005).

[17]. Pour, G., Agent-Oriented Software Engineering (AOSE): its emergence as a cornerstone of enterprise software engineering education. *World Trans. on Engng. and Technology Educ.*, 2, **2**, 225-228 (2003).

[18]. Pour, G., Component-based development refining the blueprint of software engineering education. *World Trans. on Engng. and Technology Educ.*, 2, **1,** 45-48 (2003).

[Jazayeri95] M. Jazayeri, "Component programming: a fresh look at software components," Proc. 5th *European Software Engineering Conference (ESEC '95)* (Sitges - Barcelona, Spain), LNCS 989, pp. 457-78, September 1995.

[Oberleitner02] J. Oberleitner and T. Gschwind, "Composing Distributed Components with the component Workbench," Proceedings of the 3rd

[Gschwind02] T. Gschwind, *Adaptation and Composition Techniques for Component-Based Software Engineering*, PhD dissertation, Distributed Systems Group, Technical University of Vienna, 2002.

.