A Study and Categorization of Pervasive Systems Architectures: Towards Specifying a Software Product Line

Mostafa Hamza, Sherif G. Aly
Department of Computer Science and Engineering
The American University in Cairo
{hamza, sgamal}@aucegypt.edu

Abstract

Abstract - In this work, we identify the importance of building a reference architecture for pervasive systems that could be used as part of a specification of a software product line for this domain. In an attempt to eventually specify such reference architecture, we studied over fifty published architectures for pervasive systems. We then extracted the major architectural features present in the studied architectures, and categorized the architectures according to the type and the environment they are targeting. We present our findings herein this article.

Key words: pervasive, software product line, reference architecture.

1. Introduction

The ever increasing and innovative requirements imposed upon new breeds of software systems correspondingly increase the complications of design and implementation. The domain of Pervasive Systems development is one of those domains that go beyond the “traditional” way of development. Such systems are characterized by their ability to sense surrounding context, and correspondingly react intelligently in a way that relieves users from interacting with software using unnecessary jargon. Software development methodologies for this domain have yet to reach a maturity stage that facilitates the development of such systems.

Pervasive systems are characterized by the presence of context, actors, sensors and events. Context represents ambient information. Actors are the individuals that are exposed to the system either directly or indirectly. Sensors provide contextual inputs to the system without human intervention. Finally, the events support information flow within pervasive systems.

In all pervasive systems, there are different challenges that pose themselves. According to [12], the main challenges include heterogeneity, scalability, dependability and security, privacy and trust, spontaneous interoperation, mobility, context awareness, context management, transparent user interaction and invisibility.

Current implementations of pervasive systems are still based on ad-hoc development activities, which do not fit for scalability or evolution. On the other hand, the adoption of Software Product Lines (SPL) proved to be an effective approach in obtaining successful development in a shorter time to the market with improved productivity and quality [23]. SPL works on the development of software components that share a common, managed set of features using the same set of core assets. In this paper, we gather from literature the most pressing architectural features that should be present in any pervasive software.

The paper is organized as the following: next we will talk about the conducted survey and the categorization done from the pervasive architectures highlighting the common features in each category. Finally, we recapped the article in the conclusion.

2. Surveying and Categorizing Pervasive Architectures

The main target of our research is to propose a generic reference architecture that can be used to obtain software product lines for pervasive systems. The research is divided into three phases. The first phase is to define pervasive systems and the main characteristics for achieving ubiquity and pervasiveness. The second phase is to group the different features from the various architectures of pervasive systems and categorize them. The third phase is to define the reference architecture and evaluate it.

The first phase was initiated by investigating the pervasive systems’ characteristics from previous work. We narrowed down our related work collection to focus on approaches that were adopted for defining the pervasive systems. The settled characteristics are ubiquitous access, context awareness, intelligent interaction and natural interaction as discussed earlier.

Throughout the second phase, we survey the proposed architectures for pervasive systems. So far, the collected architectures are around 50. We selected the most structured architectures and started gathering the characteristics and features in each one. The approach that all the architectures followed in order to achieve pervasiveness is to minimize human interaction. We categorized them according to their usage and operating environment. Then we extracted the common and variable features from the architectures.

The third phase will be to define the reference architecture from the categorization and feature extraction that resulted from the previous phase. The reference architecture will be generic enough in order to be applied to any pervasive system. As shown in figure 1, we categorized...
the architectures into General, Bridging, Domain Specific, Privacy, Fault Tolerance and Context-awareness.

3. Features
In this section, we describe each of the major features found in the surveyed pervasive architectures indicated in Figure 1.

3.1. General:
We extracted the general features that should be there in each and every pervasive system. The extracted features [46] are: context, actors, sensors and events.

3.2. Privacy:
We extracted the privacy features from [28], [44], [21], [8], [36], [38], [30], [6], [18], [3], [1], [2] and [32]. As shown in figure 2, privacy pervasive systems consist of Trusted Channels or Trusted Points or both for securing communication. Authentication is used to ensure the identity of connected users. Authorization and Roles is used to grant privileges for different users to access the functionalities. For Identity hiding, there are different techniques that could be used. A Proxy is used for anonymity. Pseudonyms and Dummy Traffic and Mixzones are used to conceal the users from the provided pervasive functionalities.

3.3. Learning:
As shown in figure 3, learning pervasive systems are considered domain specific pervasive systems. They are characterized by the following features which are extracted from the previous pervasive architectures [45], [5], [16], [19] and [34]. The features are:

- **Learner Profile**: To reflect the learner’s interests and motivation in order to locate instructors.
- **Learning Environment Model**: Is used to allow learners to select the physical surrounding resources that could be used in their learning process.
- **Educational Model**: Is divided into Learning Material and Agents
  - Learning Material: Is the material that is used in the learning process. It could be audio/visual or softcopies.
  - Agents: Are either Resource Agents or Q and A Agents. The Resource Agents are used to manage the resources available either physical or virtual. The Q and A Agents are used to provide the learner and the instructor a way of communication to document their interaction.
3.4. Smart Active Spaces:
As shown in figure 4, Smart Active Spaces are considered domain specific pervasive systems [25], [39], [29], [42], [1], [37] and [33]. The features of such systems are divided into Virtual Spaces, Agents and Services.

- **Virtual Spaces**: Are the hypothetical spaces surrounding the user. They are divided into Sessions and Perceptual Components.
  - **Sessions**: They are used to link the user data and applications with the users. Users can roam around different places where he/she gets his/her data and applications available.
  - **Perceptual Components**: They are used to extract context indications from collected signals.

- **Agents**: Are responsible for tracking and modeling higher-level contextual situations.

- **Services**: Are the set of services that should be available for the smart pervasive systems:
  - **Presence Service**: Collects the information about the active space resources, i.e. it keeps the status of the software components, people and devices.
  - **Media and Data Component Service**: Supplies the participants with real-time audio/video communication. (This is used in smart meeting spaces)
  - **Multimodal Interaction Component Service**: Provides user-friendly interaction with the tiled display.

3.5. Health:
As shown in figure 5, Health Pervasive Systems are one of the domain specific pervasive systems. The extracted health features from the previous pervasive architectures [11] and [40]. Health features are divided into the following:

- **Drug Manager**: Is responsible for managing the supply of drugs to the patients according to the situation and the need.

- **In-door and Outdoor Handover**: Handover mechanism to switch between monitoring indoors and outdoors to sustain availability all the time.

- **Health Sensor Network**: Contains the health sensors that monitor the patient’s situation such as heartbeat, blood pressure … etc.

- **Database**: Contains the patients’ history as well as the medication required for them to keep track of their progress.

3.6. Games:
As shown in figure 6, the extracted features from games pervasive systems [7] and [27] are:

- **Gesture Based Interaction Devices**: Are the devices used to allow the players to interact with the games using their body movements or using external devices.

- **Gesture Recognition**: Is used to capture the players’ body movement and send to the game engine the required action.

- **Pointing Devices**: Are used by the players for easily playing without being so close.

3.7. Mobile:
As shown in figure 7, the extracted mobile features from the previous pervasive architectures are [35], [4] and [14]:

- **Security**: Is an important feature in order to prevent any unauthorized access for the mobile resources.

- **Sandbox**: Is the same concept as in java. It is used to execute the downloaded programs and services in tightly-controlled resources.

- **Mobilets**: Are chain of service objects used to supply improved services to the underlying mobile applications. They can be added, updated or deleted dynamically.

- **Mobile Manager**: Is responsible for executing and/or
migrating the services and the programs.

- **Executer**: Is responsible for executing the downloaded programs inside the sandbox.
- **Service Migration**: Is responsible for the services to migrate between the mobile phones and the found resources.
- **Surrogates**: Are wired resources that mobile phones can use to filter the services and to communicate with the suitable ones.

![Diagram of Mobile features]

### 3.8. Retail:
As shown in figure 8, the extracted retail features from the previous pervasive architectures [15]. Retail features are:

- **Shopping Cart**: The shopping cart is equipped with Readers and a PDA to keep track of the added items and display to the customer the price and special offers for the related products.
- **Readers**: They are used to keep track of the products and their location. They are installed on the shopping cart and on the shelves.
  - Bar Code Reader: Is type of monitoring the products.
  - RFID Reader: Is Radio Frequency Identification to monitor the products.
- **Transcoder**: Is used to communicate with the back-end system of the shop through transforming the data and ensuring their availability to the customers.

![Diagram of Retail features]

### 3.9. Emergency:
As shown in figure 9, the extracted emergency systems’ features from the previous pervasive architectures, [31] and [20], are categorized by the following features:

- **Distance Calculator**: Is used to calculate the distance between the sensors and the neighbors as well as the nearest emergency team and the situation place.
- **Workflow Manager**: Is responsible for assigning tasks to the emergency team according to the different predefined models.
  - Workflow Execution Engine: Is used to assign tasks to the emergency team.
  - WorkflowReviewer: Is used to review the tasks given and report if they are done correctly or not.
- **Situation Context Information Management Service**: Is responsible for saving the context data related to a situation in a database for providing support for the emergency teams and for post-situation analysis.
- **Team Manager**: Is responsible for monitoring the team’s progress and prioritizes tasks.

![Diagram of Emergency systems’ features]

### 3.10. Transportation:
As shown in figure 10, the selected transportation features from the previous pervasive architectures [12] are:

- **Legacy Tier**: Is responsible for integrating with the current traffic systems.
- **Management Tier**: Is responsible for managing the incoming traffic data that are collected and analyze them.
  - Geo-data Collector: Is responsible for collecting the geographical data from the streets, filter them and send them to the management tier.

![Diagram of Transportation systems’ features]

### 3.11. Bridging:
As shown in figure 11, the collected bridging features from the previous pervasive architectures, [10] and [24], are:
• **Context Broker**: Is responsible for identity management and context discovery.
• **Context Producer**: Is responsible for handling context adaptation, reasoning, and formatting context information for foreign CMSs.
• **Interoperability**: Is responsible for exchanging information between different devices. It is divided into Mapper and Translator.
  o **Mapper**: Is responsible for creating service-level bridges for recognizing the newly appeared devices and transport-level bridges for abstracting the communication.
  o **Translator**: Establishes device-level bridge for native devices. Moreover, it is responsible for translating the different representations of device semantics as well as working as a proxy for that device.
• **Directory Module**: Is used to handle the availability of the devices.

![Figure 11: Bridging Systems’ features](image)

3.12. Context Awareness:
As shown in figure 12, the extracted context-aware features from the previous pervasive architectures, [26] and [17], are:
• **Adaptation Manager**: Is repositories for situation, preference and trigger and evaluate on behalf the application layer according to the results of the query layer.
  o **Situation Repository**: It contains all the situations and the changes happen to them i.e. context changes.
  o **Preference Repository**: It holds the preferences for each user.
• **Context Manager**: Is responsible for maintaining a set of context models for the applications to contact.
  o **Model**: Is used to support the different tasks that can be carried out by the users.
  o **Context Repository**: Maintains all the models extracted.
  o **Context Wrapper**: Is responsible for encapsulating particular or singular context information to be supplied to the Context Reasoner.
• **Context Reasoner**: Is responsible for interpreting collected context information from the sensors and filtering them according to analysis techniques.

![Figure 12: Context-aware features](image)

3.13. Fault Tolerance:
As shown in figure 13, the extracted fault tolerance features from the previous pervasive architectures, [9] and [41], are:
• **Checkpoint Store**: Is responsible for storing the status regularly of all the devices and the sensors connected.
• **Fault Manager**: Is responsible for managing the applications and devices when they got disconnected and search for the next available application and device to failover to.
  o **Heartbeat Messaging**: Is used to ensure that the applications are connected.
  o **Fault Notification**: Is used to notify the fault manager when any device or application is got disconnected.

![Figure 13: Fault Tolerance features](image)

3.14. Data Migration:
As shown in figure 14, the extracted Migration features from the previous pervasive architectures [21] are:
• **File Manager**: Is responsible for managing migrating files on the move by abstracting the location of the file without the interaction from the user.
  o **Cache**: Is used to cache the files on the move in order not to lose them, i.e. redundancy.
  o **Migration Queue**: Is a data structure responsible for uniting the file manager’s outbound communications.
3.15. Document Editing:
As shown in figure 15, the collected document editing features from the previous pervasive architectures [43] are:
- **Document Editing Tools**: The tools are used to edit the documents. They communicate with the real-time server components to reflect the changes automatically.
- **Database**: Is responsible for saving the documents.
- **Real-time Server Components**: They are used to propagate the information to all the connected users.

![Document Editing features](Image 15)

4. Conclusion
Throughout the paper, we have presented the proposed categorization and their features from the pervasive systems’ architectures. This will help in generating a reference architecture to be used for yielding a software product line for pervasive systems. Currently, we are working on generating the complete feature diagram with the almost all the major capabilities a pervasive system could have. The next step will be generating from each feature the set of components required for a feature to operate. Finally, we will be combining them in one comprehensive reference architecture that can be configured to produce specific pervasive system architectures.

5. References


