Pebbles and Urns: A Tangible, Presence-Based Service Delivery Framework

William Otho Plymale

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Engineering

December 14, 2012
Blacksburg, Virginia

Keywords: pervasive computing, opportunistic, situated

Pebbles and Urns: A Tangible, Presence-Based Service Delivery Framework

William Otho Plymale

Abstract

Wireless and pervasive computing research continues to study ways the Internet of Things (IoT) can make lives easier and more productive. Areas of interest include advances in new architectures and frameworks that support large-scale IoT deployments beyond research prototypes, simple and inexpensive human-to-device and device-to-device interfaces, and user decision making support with opportunistic information services.

This dissertation investigates the design and implementation of a general-purpose framework upon which IoT and opportunistic computing (OC) systems can be built.

The result of this work is Pebbles and Urns (P&U), a casually accessible system designed to deliver information to a person that is pertinent and beneficial to them with respect to their current activity, location and other contexts. P&U is a proximity-based information delivery framework that leverages a simple, inexpensive tangible interface and context-rich, physically-situated, distributed information repositories. By its proposed use of enforced proximity, local context, and location-specific services, P&U can support the situated interaction between user and place.

The P&U framework is based on a layered architecture consisting of an isolated physical communication layer, a data repository supporting opportunistic service composition and delivery, and a controller/interface providing user feedback. Serving as a potential IoT design pattern, P&U application developers can use the framework API's and software tools to build and deploy P&U systems.

As validation of this work, P&U prototypes are constructed using the framework, API's and software tools. The prototypes are based on use cases depicting a person engaged in the day-to-day activities of attending class, going to the gym and grocery shopping. Performance measurements are performed on the prototypes profiling core components of the framework. Results indicate proper functioning of P&U tangible interfaces, communication connections, service request and delivery, and internal framework operations.

Contributions of this research include a general-purpose framework, a simple IoT interface and an opportunistic engine.

Acknowledgements

Completing this dissertation marks the end of a long journey. As the traveler, I was and am defined by three events: the launching of Sputnik 1, science fiction and the "circle."

I was five years old when the Soviet Union launched the first artificial Earth satellite. The launching of Sputnik 1 began the Space Race. The US reacted to this technological feat by creating ARPA, NASA and the National Defense Education Act. The federal government began pouring unmatched amounts of money into science education, engineering and mathematics at all levels of education placing a new emphasis on science and technology in American schools. I remember the tests; aptitude and achievement tests administered throughout my time in elementary school. During the course of this testing, I was officially labeled an "underachiever". Performing less than expected by my teachers, I tended toward procrastination, distractions and wandering. I would excel on occasion, but just barely.

I read my first science fiction book while in the second grade. It was named "Zip-Zip and his Flying Saucer". I was hooked with Heinlein, Clark, Asimov, Bradbury, Niven, Vonnegut, Seuss, Doc Smith and others becoming my new companions. I read "The New Tom Swift Jr. Adventures" instead of the Hardy Boys. I remember wanting to be like Tom Swift, teenage scientist saving the world with his marvelous inventions, and still taking time to help his mother wash the dishes after the evening dinner. To me, science fiction is fiction only for the moment.

The "circle" began when I entered college, accepted as "early decision" into the electrical engineering program. Over the next five years, underachieving at a remarkable level, I spiraled through changes in major from electrical engineering to computer science to psychology, finally managing to graduate. My first job was a freight "cracker" at a local department store. I eventually realized I very much needed to climb out of this particular rabbit hole. I managed to

get a job working “midnight shift” as a computer operator at a local furniture company. I started taking data processing classes at a nearby community college. I desperately wanted to return to the college where I spent those embarrassing five years. I wanted to atone for my failures. The “circle” continued as I retraced and back-tracked previous steps. I got a job back at the college. Working full-time, I started a second undergraduate degree in computer science. I completed it in six years. Armed with this new CS degree, I left the university in search of a “real” job. I found one, but for reasons including continued pursuit of the “circle,” I returned to the university, assumed another fulltime job, and started a master’s in electrical engineering. This one took ten years to complete, but I did it. By that time, academia and research were in my blood. Never quite fitting into my professions, often a target of re-definement by others, I was driven to complete the “circle”. I was accepted into computer engineering’s doctoral program. Still distracted and a wanderer, my progress was often at a standstill. However, 55 years after Sputnik 1, I approach completion of this degree and my “circle.” I still procrastinate, I still honor distractions and I still wander. But this dissertation process has made me realize that these are not my weaknesses, they are my strengths. My distractions consist of journeys down paths with no destinations and no intentions. I always return with something useful, perhaps not at the time of return, but certainly later in the future. I wander, but I wander with purpose, collecting thoughts and experiences that I instinctively know I need. And now, like Tom Swift, I feel like a scientist.

Despite, and because of, my “strengths,” it would have been impossible to complete this dissertation and PhD process without the help and support of others.

Two people understood what it would take for me to finish, and their belief and support of me was unrelenting.

Simply put, this dissertation would not have been possible without the support, guidance and patience of my advisor and friend, Dr. Scott Midkiff.

My wife, Patricia, has been proud and supportive of my work. She has shared the many uncertainties, challenges and sacrifices of completing this dissertation, especially during the last 15 months. This work is my gift to her. Now we can play.

Each member of my committee has inspired and directed me on this journey. Dr. Tom Martin introduced me to pervasive computing. It would take me days to decipher HCI and CSCW topics discussed during conversations with Dr. Deborah Tatar. Dr. Luiz DaSilva asked me the "killer app" question during my qualifying exam. A question I think about almost every day. Dr. Joe Tront, in handing back my first VLSI test, commented "this is not your area of interest, is it?"

The following individuals "saved" me during my traversal of the "circle".

Mrs. Madeline Key was my 9th, 11th and 12th grade science teacher. Even though I was not her brightest student, she encouraged me to become president of the high school scientist club. A role I assumed for two years. She instilled my interest in science and I am forever grateful.

Dr. Robert Frary was my first mentor as I started out the rabbit hole. He taught me sound concepts pertaining to so many areas including academics, finances, and good food and wine. I adopted a good portion of Dr. Frary's view of the world.

Dr. George Gorsline, CS professor, was instrumental in my achieving my undergraduate computer science degree. He specialized in helping students like me.

I was the only CS student in Dr. Charles Nunnally's EE microcomputer design course. I learned a number of funny CS jokes from the EE students in the class. I used two weeks of annual leave completing the lab assignments. "Power, data and control," Dr. Nunnally.

Other individuals to whom I am thankful include my sister, June. Her sacrifices during the past years enabled me to finish this degree. Priorities and decisions can sometimes be difficult and cold.

My stepson Warren is an “anti-underachiever”. I will always admire his study habits, work ethics and resulting successes. Now it is his turn for another degree.

In closing, I would like to remember Edward McPherson. He was a friend and coworker who demonstrated passion and vision for his interests. Our conversations challenged me. I had hoped we could have shared our experiences in pursuit of our respective degrees.

All photos by author.

Table of Contents

Figures

Tables

Chapter 1: Introduction

This document proposes a new pervasive computing conceptual architecture called Pebbles and Urns (P&U) and describes preliminary and expanded research based on this new architecture. To explore the feasibility and utility of the P&U conceptual architecture, an instance of the P&U architecture is investigated, designed, and implemented. This P&U instance, or framework, is used as the foundation for a tangible, situated opportunistic computing system. Feasibility studies are performed on using this P&U framework to build application prototypes. Factors relating to P&U components are identified and discussed. Finally, findings based on the P&U framework and prototypes are used to assess the fulfillment of research objectives.

1.1 Background and Motivation

The concept of the Internet of Things (IoT) became popular in 1999 when the Auto-ID Center at the Massachusetts Institute of Technology (MIT) designed Radio-Frequency Identification (RFID) technology. Kevin Ashton, co-founder and director of AutoID Labs, in an article published in Forbes Magazine, in 2002, said [1]:

> "We need an 'Internet-for-things,' a standardized way for computers to understand the real world."

At the same time, researchers at MIT's Media Lab referred to the IoT with respect to the World Wide Web [2]:

> "In retrospect it looks like the rapid growth of the World Wide Web may have been just the trigger charge that is now setting off the real explosion, as things start to use the Net."

The IoT has evolved to represent a physical world saturated by fixed and portable devices with computing and communication capabilities. The IoT provides a foundation for connecting things, sensors, actuators, and other "smart" technologies, thus enabling person-to-object and object-to-object communications. While an area of continuing research interest in the wireless and pervasive computing domain, there is no clear understanding of how the IoT will make our lives easier and more productive. Currently, it is perceived as affording information visibility. Researchers claim the IoT needs to support autonomous decision making to reduce everyday decision tasks and avoid delays between information availability and decisions [3].

With the proliferation of networked devices connecting people and things, the amount of accessible, real-time information enabling choice and decision making will increase dramatically. A new IoT "ecology" consisting of the relations and interactions between people, these devices, and the environment is expected to form. Devices in this ecology will be based on new technologies, as well as traditional computing, networking, and storage devices. As connected devices number in the billions, approaching trillions, there is a realization that current computer and network technologies are unable to handle the complexities of interaction associated with this volume of devices [4]. New service migration and coordination infrastructures are needed to support the types of services offered in this new "ecology." The real-time data storage and retrieval needs of this large number of devices will require new distributed data storage and shared memory models. These numerous heterogeneous devices will possess varying communication, computation, and network abilities, and varying levels of reliability. In addition, the nomadic behavior of users will add interaction complexities. Systems of unstable networks and devices will require new, robust, scalable network and messaging models. New human-to-device and device-to-device interfaces are also needed. Consideration of simple, dedicated personal devices instead of multi-purpose devices like smartphones is one motivation for this research. Although successful IoT research prototypes have been developed, advances in new architectures and frameworks are needed to support large-scale deployment of the IoT. Researchers from the Institute for Pervasive Computing summarize the technical challenges associated with the underlying technologies of the IoT [5]:

1. Scalability – Communication and service discovery components of the IoT need to function in large-scale environments.

2. "Arrive and Operate" – Interactions between humans and devices in the IoT require spontaneous connection and organization.
3. Interoperability – The diversity of many different smart objects require common practices and standards.
4. Discovery – Services for "things" must be automatically identified.
5. Software complexity – Software infrastructures are needed to manage smart objects and provided services.
6. Data volumes – "Real-world" awareness scenarios require volumes of local and remote data.
7. Data interpretation – The interpretation of local context to trigger further action is not trivial.
8. Security and personal privacy –Personal privacy within the IoT requires support of selective access control of services and information among users and the many smart objects.
9. Power supply –The many devices in the IoT need to take advantage of energy saving advances in hardware and software.
10. Interaction and short-range communication – Development of new interaction techniques and processes between humans and devices in the IoT is needed.

Within the world of the IoT, intentional and random encounters between users and devices become opportunistic. In fact, the frequent likelihood of opportunities is a motivation for this research. Humans carry mobile devices, and human mobility generates communication opportunities when two or more devices come into contact. When two devices come into contact, an opportunity occurs to share and exploit each other's software and hardware resources, exchange information and execute tasks remotely. This interaction, referred to as opportunistic computing (OC), is considered one of the next major challenges in pervasive computing [6].

The design and implementation of an OC system offers several challenges. One challenge is how to manage the stability, duration, and frequency of the often wireless connection between OC devices. Connections among mobile devices are intermittent, short in duration, and suffer from insufficient, or unavailable, wireless infrastructures. Another challenge is resource availability and discovery. How are appropriate resources identified and used in opportunistic meetings? Researchers acknowledge the difficulties in defining a standard composition of services that would be available during OC sessions. Creating incentives, or reasons for initiating device conversations, is a difficult challenge given the random encounters of multiple mobile devices. OC systems include a social awareness component based on context management describing user behavior within social groups. Context representing human behavior can consist of current user location, social links with other users, and user activity within a place. This is another aspect of OC that is extremely difficult to implement and manage due to the multiple mobile entities involved. Finally, the challenges associated with power management of mobile devices are ever present.

1.2 Research Objectives and Questions

As a contribution to this research domain, a new conceptual architecture called Pebbles and Urns (P&U) is proposed to better understand and address some of the key needs described above. Pebbles and Urns is, at its core, an information delivery system that explicitly supports context-rich, physically-situated, distributed information repositories.

Urns are associated with a specific, physical location and represent a context and content repository pertaining to that physical location. Urns provision services and information specific to their physical location and of interest to mobile users nearby. Information consists of content relative to the physical location and, perhaps, other context. For example, information could include “Today’s Specials” if the location is in or near a restaurant. A portion of the content at an Urn can be information generated through user interaction. For instance, keeping with the dining theme, information could include a log of menu item selections by different users. An urn’s information may also contain a context portion of real-time environmental data about the place, such as current noise level in the dining area.

Pebbles are a tangible means of conveyance for an information-bearing message sent to or received from an Urn. Pebble information can be a query message or service request, information returned from one of those interactions, or even an executable program. A user exchanges Pebbles with an Urn through various forms of physical, situated interaction. The Pebble may be a tangible object physically placed in or removed from the urn by the user. The Pebble might also be a virtual object, for instance if the Urn is accessed by a command protocol issued by the user from his or her mobile phone or other mobile device. Urn interactions might be in the form of pointing or scanning actions performed by the user with their mobile device. Regardless of the physical form of the Pebble itself and the mechanism for Pebble interaction with an Urn, a Pebble is conveyed to or from an Urn only through physical proximity. Thus, while the Pebble and Urn may be physical or virtual, the P&U architecture is predicated on enforcing physical proximity and enriching the interaction through context.

There are usually lessened power concerns for the Urn since it can typically be powered by a continuous power source. Also, the Urn is expected to be connected to a reliable and fast, perhaps wired, network infrastructure. This reliable network component facilitates access to external service. Pebble resources are expected to be less than those of an Urn. This discrepancy in resources provides an opportunity to explore the use of cyber foraging strategies in Pebbles and Urns. Cyber foraging is the transient and opportunistic use of computing resources by mobile devices [7].

Given the challenges of the Internet of Things and opportunistic computing presented in Section 1.1, research questions (RQ) associated with the proposed new P&U architecture to be addressed in this research are as follows.

RQ 1. How would properties and characteristics of the P&U architecture advance the design and deployment of IoT and opportunistic computing implementations? Guiding components of this research question include the following.

RQ 1.1 What interoperability and interaction features and characteristics of the P&U architecture provide a stable and flexible communication connection supporting near, situated, tangible interaction between devices participating in the application domain of opportunistic computing?

RQ 1.2 Within the P&U architecture, what protocols, rule sets, algorithms, abstractions, and context management identify a relevant service domain for parties participating in a P&U-based opportunistic computing session?

RQ 1.3 How can P&U designs utilize cyber foraging strategies such as power management and cloud services?

RQ 2. What innovative, or problematic, features are revealed through the design and implementation of a P&U instance and, for problematic features, can the problem areas be addressed?

Research Question 1 is the primary question that this research attempts to answer. Pertaining to technical challenges associated with the underlying technologies of the IoT and OC, RQ 1.1, RQ 1.2 an RQ 1.3 identify specific research areas that were examined for key P&U properties and characteristics. Research Question 2 addresses areas explored during the course of this research and in future research. In addressing these questions, new questions presented themselves and, in turn, were investigated as time permitted.

1.3 Overview of Research Methodology

Several high-level design models of different Pebbles and Urns instances were defined. Analysis of these models were performed to determine how closely each represents the Pebbles and Urns conceptual architecture, described in further detail in Chapter 2 of this document.

To answer Research Question 1 presented in Section 1.2, prototypes based on the Pebbles and Urns architecture were constructed. Guided by Research Questions 1.1, 1.2 and 1.3 presented in Section 1.2, these prototypes were used to explore and instantiate the contributing utility of a Pebbles and Urns system relative to unique, reusable OC structures and properties. Experimental evaluation of this prototype includes a feasibility study identifying implementation considerations and performance factors associated with Pebbles and Urns system components.

New questions arose as a result of this experience. Questions that, when explored, will contribute to a better understanding in answering Research Question 2 in Section 1.2.

1.4 Anticipated Contributions

This research has led to the proposed Pebble and Urn architecture. While the full realization and investigation of all aspects of Pebbles and Urns is beyond the scope of this dissertation, this research explores ways in which novel elements of a Pebbles and Urns system can assist in the design and implementation of Internet of Things and opportunistic computing applications. In a P&U system, one node, the Urn, is stationary. Pebbles represent the many mobile or nomadic users and their devices. Pebble and Urn interactions remain opportunistic given the dynamic nature of user intent and the ever-changing context of user and place. The Urn is situated in a physical place and represents that place in terms of consistent, trusted, well-known services and information. The social context of an OC system is managed by the Urn, as it maintains a cache of information "residue" pertaining to its physical place [6]. Location context is a constant and social behaviors are generally defined and bounded by the nature of the place associated with its Urn. Contributions, as indicated by the Research questions in Section 1.2, are the studies of interoperability, service domains and cyber foraging strategies within the framework of a new P&U architecture that is used to better understand and address IoT and OC implementations.

Another contribution of Pebbles and Urns is the support of information conveyance in the approaching new "ecology" of the Internet of Things. The Pebbles and Urns architecture supports systems that augment the numerous ways a person deploys, obtains, and realizes information in their daily activities. Types of interactions include the human-to-device relationship of a person and their Pebble, the device-to-device communications between Pebbles and Urns, and the human and device communications with supporting pervasive services. Pebbles and Urns services include context management, personalization, service discovery and composition, session management, UI mechanisms, and privacy and security features.

1.5 Document Organization

This chapter has discussed the background and motivation for the design of a pervasive computing conceptual architecture called Pebbles and Urns. Chapter 2 covers the core elements of the P&U conceptual model including descriptions of a Pebble and an Urn, P&U interactions and information flow, Pebble and Urn repositories, and Urn services. Chapter 3 provides an overview of background and prior work contributing to the P&U architecture. Design models of a P&U architecture are described and analyzed in Chapter 4. Chapter 5 details the design, implementation and testing of a minimalist P&U instance based on the initial P&U architecture discussed in Chapter 4. Chapter 5 next presents research based on this early P&U work, and its application to the research questions presented in Section 1.2. Finally, Chapter 6 summarizes the research discoveries and contributions of this study and identifies areas of future work.

Chapter 2: Pebbles and Urns Conceptual Model

As presented in Chapter 1, Pebbles and Urns is a tangible, presence-based messaging architecture that can serve as a foundation for situated opportunistic computing systems. This chapter describes the core elements of the base P&U architecture. Section 2.1 provides a conceptual overview of the P&U architecture. P&U core components, including the Pebble, the Urn and associated information repositories, are introduced in Section 2.2. Section 2.3 describes aspects of Pebble and Urn interaction and communication. A discussion of situated user services delivered by an Urn is presented in Section 2.4. Use cases demonstrating the potential utility of the P&U architecture are described in Section 2.5. Section 2.6 summarizes the P&U topics covered in this chapter.

2.1 Pebbles and Urns Concepts

Relevant is defined as pertinent to the matter at hand [8]. The P&U system is designed to deliver relevant information, computation, and communication that is pertinent and beneficial to a person with respect to their current context, including their activity and location. Their personal experience is enhanced by this additional information. P&U is intended to be casually accessible, supporting situated interaction with its use of locally-generated content, contextual data, and distributed services. P&U can meet the goals discussed in [9] by embedding interaction and computation deeply into the natural flow of everyday tasks, activities and collaborations. The elements of P&U that enable this utility include a natural user interface, a situated component, a strong physical association between user and activity, up-to-date content about the user, location, and timely services.

The P&U user interface involves the association of a token, the Pebble, owned and carried by the user with a receptacle object, the Urn, fixed to a specific place or physical location. The user may be represented by their Pebble, at least at that instant of time and for the user's intended purpose. A user may possess several different Pebbles, each representing them in a different role. The Pebble contains information about the person that facilitates activities within their environment. Examples of this information could include user identification data, content pertaining to upcoming activities and other information. The Urn receives the user's Pebble, initiates an exchange of Pebble and Urn identifying information, and then establishes a connection or association between the user and information services relevant to the user's current activity. Urns can be distributed and located throughout the relevant portions of a user's environment, including their home, workplace, shopping locations, and/or recreational locations. Places in the environment are embodied by Urns at those places. An Urn represents a place by containing information about its physical location, nature of business, environmental characteristics, and current state. The current state of a place describes its attributes. For example, a possible attribute of a restaurant is the number of currently available seats. An Urn has a strong physical association with the place or location that it represents. An Urn is a focal point of information about that place. It is that location's omphalos, where omphalos is defined as a central part or focal point. Omphalos is the name of the rounded stone in the shrine at Delphi, regarded by the ancients as the center of the world [10].

People move about their daily activities following both established patterns and random deviations caused by choice, decision making, casual encounters and social behaviors. Given the mobility of the user and their pebble and the distribution and association of Urns with numerous places of interest, Pebble and Urn interactions will occur. With its computational, informational, and service composition capabilities, the P&U architecture assists with the formation of opportunistic user and place interactions.

P&U leverages a simple, inexpensive tangible interface and context-rich, physically-situated, distributed information repositories. A result of this design is that much of P&U's underlying infrastructure and operation are hidden from the user. Privacy issues arise regarding the nature of personal information exchanged between a user's Pebble and an Urn. Mark Weiser stated that "the real problem [associated with ubiquitous computing] while often couched in terms of privacy is really one of control" [11]. Controls within the P&U system that help manage user roles and privacy includes:

- Pebble Repository data structures that support master and place-specific user profiles allowing the segregation of user profile information

- Multiple Pebbles per user each representing different user roles
- Encrypted QR Pebble
- Role support on Smart Pebble

2.2 Pebbles and Urns Core Elements

This section introduces the core elements of the P&U architecture. They are the Pebble, the Urn and their respective information repositories.

2.2.1 Pebble

As introduced in Section 1.2, a Pebble is a form of an information-bearing message sent to or received from an Urn. Examples of Pebble instances are illustrated in Figure 1.

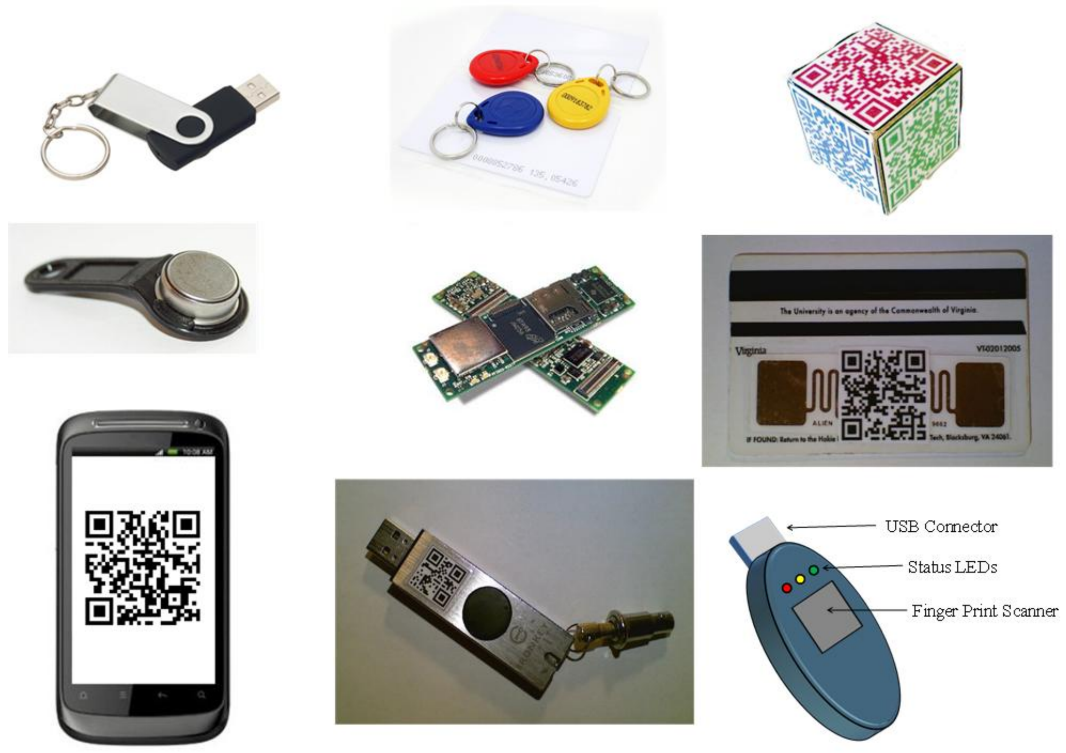


Figure 1 - Examples of Pebbles

Pebble information can be or represent a query message or service request, information returned from one of those interactions, or an executable program. A Pebble may be treated by a user in the same manner as their keys, wallet, or eyeglasses in that a user's Pebble or Pebbles are always with them. Alternatively, possession of a Pebble may be transient being associated with temporary state or situation. Storage elements of a user's Pebble can contain digital identity certificates, program executables, and other types of information relating to their current activities and behaviors. The Pebble enables user interaction with a physical location and associated services via an Urn.

Pebbles possess varying capabilities based on their underlying implementation technologies. A Pebble classification system based on different Pebble implementations defines particular Pebble and Urn interactions. A Pebble may be implemented using RFID or 2D code technologies. The physical form factor of such a Pebble is a small object containing an RFID tag or displaying a two-dimensional (2D) code such as a quick response (QR) code. This type of Pebble is read-only and incapable of receiving updated information from the Urn with which it is interacting. Another Pebble form factor is the ubiquitous Universal Serial Bus (USB) or thumb drive. A Read/Write Pebble based on a USB drive can have considerable storage capacity and can be updated with information from the Urn. More sophisticated "smart" Pebbles with UI interfaces and computation capabilities can be implemented with small embedded microcontrollers like the Gumstix Computer-on-Module [12]. For instance, an Urn interacting with a "dumb" RFID Pebble capable of only identifying itself would assume greater responsibility for areas of storage and service management. A near-field communication (NFC) interface can be based on Bluetooth or radio frequency identification (RFID). Table 1 lists implementation platform characteristics and benefits of several Pebble classes with respect to cyber foraging, form factor, communication interface, storage and Urn dependency.

Table 1 – Platform Aspects of Pebble Classes

P&U Platform	**Pebble Class**		
	Read-Only	Read/Write	Smart
Cyber Foraging	No power requirements	No power requirements	Power required (processors and radios)
Example Form Factor	Small physical tokens using QR codes or RFID tags	USB (thumb) drive	Embedded microcontroller or personal mobile device
Communication Interface	One-way technologies • Scanning • Touching	Two-way technologies • Touching	Two-way technologies • NFC • Hybrid
Typical Storage	2K to 4K bytes	128 GB	128 GB
Urn Dependency	Significant	Moderate	Peer relationship

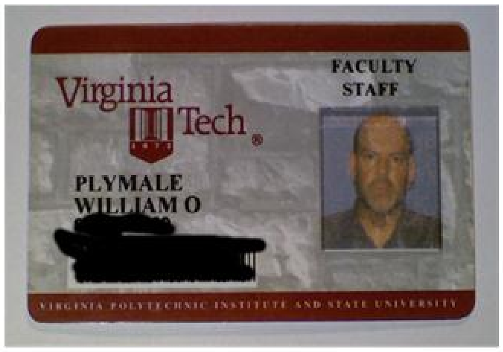
Table 2 lists architectural characteristics of the Pebble classes with respect to session initialization, storage structure, message format, physical link isolation and Pebble decision making.

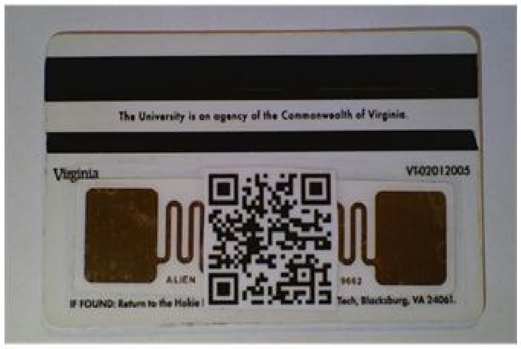
Table 2 - Architecture Aspects of Pebble Classes

P&U Architecture	**Pebble Class**		
	Read-Only	Read/Write	Smart
Session Initialization	Blocking	Device event notification like the Linux udev process	Service discovery
Storage Structure	• Simple text strings • Limited storage • Read-Only	• File directories • Ample storage • Read /Write by Urn	• File directories • Ample storage • Read /Write by Pebble and Urn
Message Format	• Simple • Constrained by Storage size	• More sophisticated • Similar to USB storage capabilities • Urn (OS) access	• Similar to Read/Write • Opportunities for active Pebble/Urn dialog
Physical Link Isolation	Wrapper code around PMI technologies handles one-way communications	Wrapper code around PMI technologies supporting bi-directional communication	Wrapper code around PMI technologies including NFC
Urn Interaction	Urn pulls	Urn pulls and pushes	Urn/Pebble pulls and pushes
Pebble Decision Making	Restricted to static information in message (text string)	Urn interactions via R/W capability of Pebble.	Yes, given presence of Pebble Controller

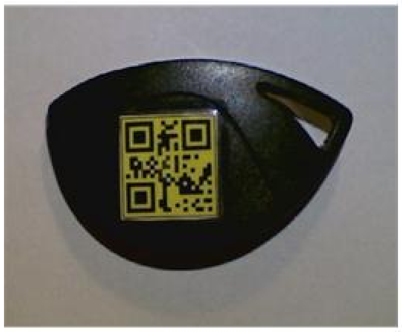
An extension to Pebble classes is a Pebble incorporating multiple physical-mobile interaction (PMI) interfaces [13]. Figure 2 shows three examples of a hybrid Pebble: an RFID keychain dongle with added QR code, a USB drive with added RFID tag and QR Code and a card form factor with magnetic stripe, QR code and RFID tag. A hybrid Pebble is capable of performing more sophisticated actions while still maintaining a minimal resource configuration.

Multiple interfaces on a single Pebble form factor can associate separate external services by combining separate pieces of information. For example, the QR code on the ID card might present the combined information contained on the magnetic stripe and the RFID tag.

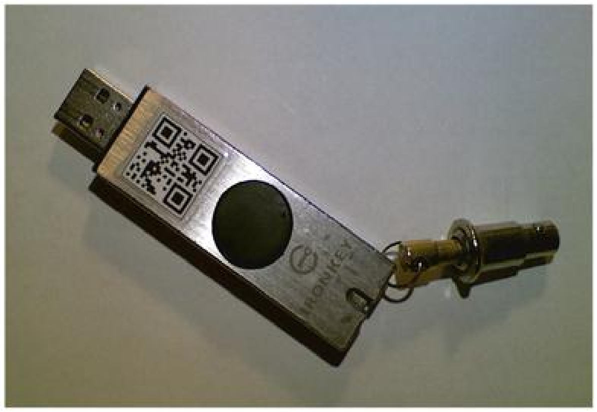



The University is an agency of the Commonwealth of Virginia.
Virginia
VI02012005
ALIEN
IF FOUND: Return to the Hokie
Tech, Blacksburg, VA 24061.

Card Form (QR Code, RFID, Mag Stripe)



RFID Keychain Dongle (QR Code)



USB Drive (QR Code, RFID)

Figure 2 - Examples of Hybrid Pebbles

2.2.2 Urn

An Urn is a physical object that is a receptacle of Pebbles. Examples of Urns are illustrated in Figure 3. An Urn is anchored to a specific physical entity and its location. An entity can be a store, a sporting arena, a restaurant, a classroom, or any space encountered by a person going about their daily activities. The Urn represents that entity. The user placing their Pebble in an Urn facilitates the user's embodied interaction with the corresponding physical space. User proximity to the Urn is required for the interaction to occur. Characteristics common across Urns include local computational and storage capabilities, some form of on-board environmental sensors, Pebble store and retrieve mechanisms, and user interface elements like visual displays and indicators, printers, and audio devices. Urns may be connected to a network or may be stand-alone. Numerous urns can be situated across an environment, each associated with a physical place. The Urn located in the user's home serves as a platform for managing the user's Pebbles by loading and updating user profile and service request information.

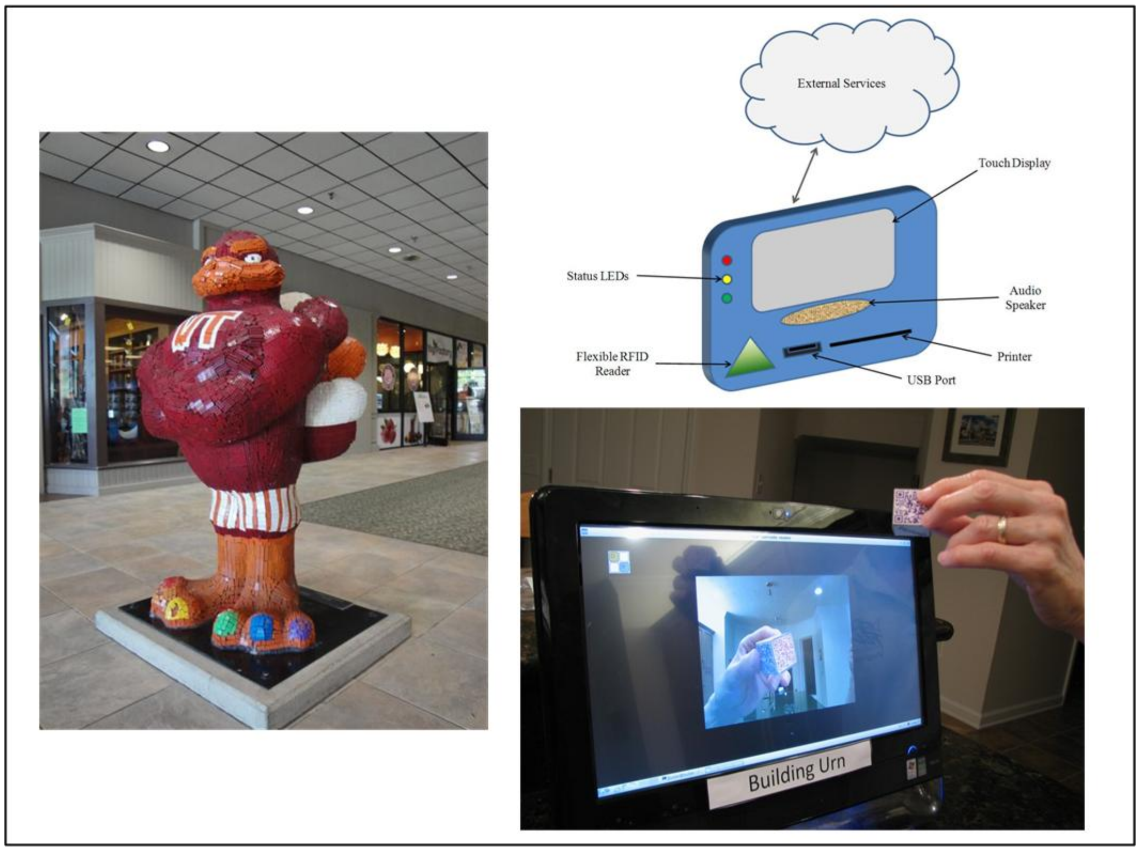


Figure 3 – Examples of Urns

One service consideration of these urns is whether or not they share information with each other. An urn can be "standalone," unconnected to other urns, managing its own content and context repositories. Advantages of "standalone" urns include no network infrastructure requirements, an enforced physical presence with place, and an ad-hoc, user-based message transfer mechanism from urn to urn. Advantages of connected urns are coordinated information transfers to multiple locations, access to non-place specific information via cloud services, and reduced local processing and storage resource requirements. This concept is expanded upon in Section 2.2.3.

Information stored in a Pebble can be read, and updated by the Urn containing it if the Pebble has writeable storage. Urn interactions enhance user experiences by incorporating contextual information.

2.2.3 Information Repository

Both Pebbles and Urns have information storage and management capabilities. Different Pebble classes have different storage characteristics. If a Pebble is capable of storing data, its Pebble Repository contains objects associated with the user's anticipated or opportunistic activities. Information stored on the Pebble can consist of data files such as a grocery list, a user's current prescription drug regimen, or service and program references. RFID-based and similar Pebbles possess limited amounts of read-only storage. Such a minimalist Pebble containing only a unique identifier would have its data store functions managed by an Urn.

Information about the Urn's place and user interactions is contained in the Urn Repository. The kinds of information and data in the Repository differ depending on whether the Urn is standalone or federated. A standalone Urn is not connected to a network and manages local information in its repository. Even when not network-attached, standalone Urns offer unique service opportunities because their local Repositories are updated only by the nomadic activities of their users. Federated Urns are network-attached and have access to external services. This concept is further explained when describing types of Urn services in Section 2.4. The Urn Repository of a standalone Urn contains:

1. local place-specific content;
2. user-generated content (activity history and/or social cache);
3. local environmental (sensor) data;
4. program executables implementing place-specific services; and/or

5. state information about Pebble and Urn interactions.

In addition, the Repository of a federated Urn contains information harvested from other Urns, either directly or via external services. Federated Urns could provide place and user information selected by place type or geographical location. This type of information access supports opportunistic situations through context awareness of nearby locations. For example, a person using a P&U instance to shop for clothes may be made aware of a lunch special at a nearby restaurant.

An Urn Repository manages several sources of user-generated content. One source is the content generated by users of the system. Information stored on the user's Pebble is collected and stored as part of the Pebble-Urn interaction. This user-generated content is further supplemented by actions performed by the Urn when the Pebble information is processed. For example, a user places a Pebble containing their grocery list in the grocery store Urn. Additional local content is generated by the Urn as it updates its "demand" content by the items requested on this user's grocery list. The other source of local content is the data produced by the various environmental sensors attached to an Urn.

2.2.4 Pebbles and Urns Architecture

Depending on the Pebble classes, the Pebble architecture consists of some combination of a Pebble controller, a Pebble repository, a P&U protocol/interface, and a physical communication layer. These components are described in more detail in Chapter 4. The Urn architecture, consisting of an Urn controller, an Urn repository, a P&U protocol/interface, and a physical communication layer, is described in more detail in Chapter 4. Figure 4 shows a high-level view of the P&U architecture. P&U interactions shown in Figure 4 are discussed in Section 2.3.

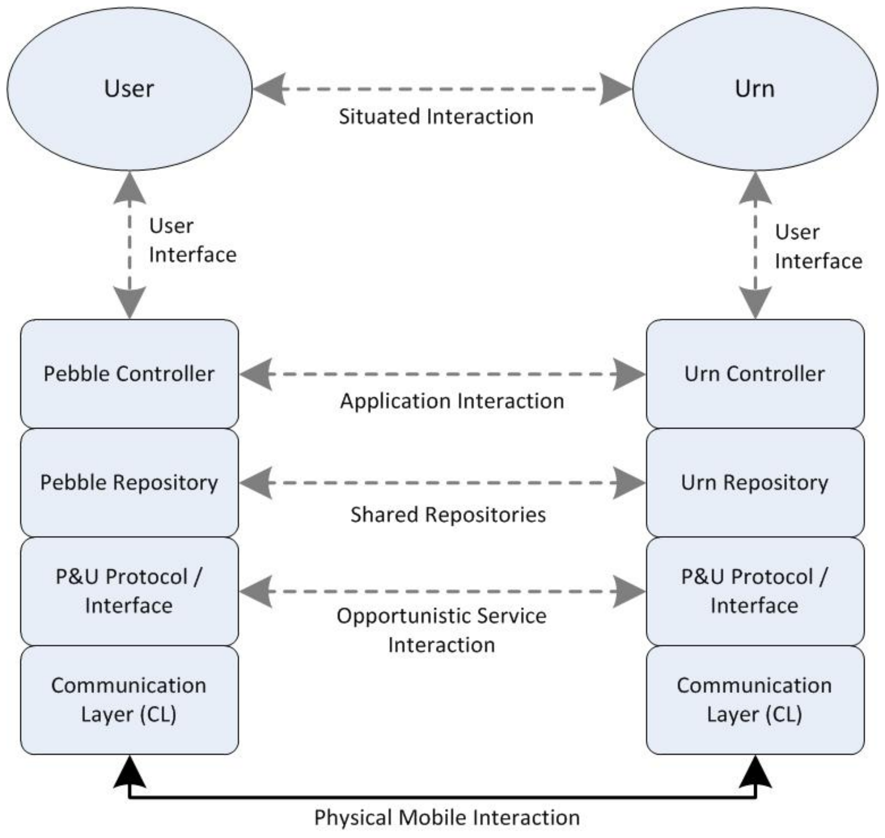


Figure 4 - P&U Architecture Components

The term "pattern", in software engineering, describes a proven solution to a common problem in a specific context [14]. Design patterns have been studied widely in the field of software and are known to improve the transfer of knowledge [15]. The use of design patterns can help software designers to design better software. Design patterns provide a common vocabulary for computer scientists across the domain barrier and enhance the documentation of software [16]. The overall architecture of a system and related design decisions can be explained by giving a set of patterns used.

P&U protocols and application programming interfaces (API) are comprised of a mix of current architectural patterns. Another aspect of this research, driven by Research Questions 1, 1.1, 1.2 and 2 presented in Section 1.2, is the investigation of layered protocol stacks that lead to a P&U hourglass-shaped or "thin waist" architecture [17] .

2.3 Pebbles and Urns Interactions

This section examines the interactions between Pebbles and Urns. This situated, physical interaction between the Pebble and an Urn is a distinguishing feature of the P&U system and is examined based on Research Questions 1 and 2. Reasons for this type of interaction include the following.

- Real-time, physical adjacency affords the most accurate information about a place.
- As a requirement of interaction or information exchange, the user or pebble must be physically present at a place, for example to pick up a ticket for an event or meet with a physician.
- P&U interaction involves physical activities specific to location, for example to unlock a door with a key.
- Interaction requires, or is facilitated by, characteristics of short-range communication (PMI), for example as might be used for a security scheme.

A user associates Pebbles with an Urn through various forms of interaction. The Pebble may be a tangible object physically placed in or removed from the urn by the user. The Pebble might also be a virtual object, for instance if the Urn is accessed by a command protocol issued by the user from his or her mobile phone or other device. Urn interactions might be in the form of pointing or scanning actions performed by the user with their mobile device. Regardless of the physical form of the Pebble itself and the mechanism for Pebble interaction with an Urn, a Pebble is conveyed to or from an Urn only through physical proximity. Thus, while the Pebble and Urn may be physical or virtual, the P&U architecture is predicated on enforcing physical proximity and enriching the interaction through context. By its proposed use of enforced proximity, local context, and location-specific services, P&U can support the situated interaction between user and place.

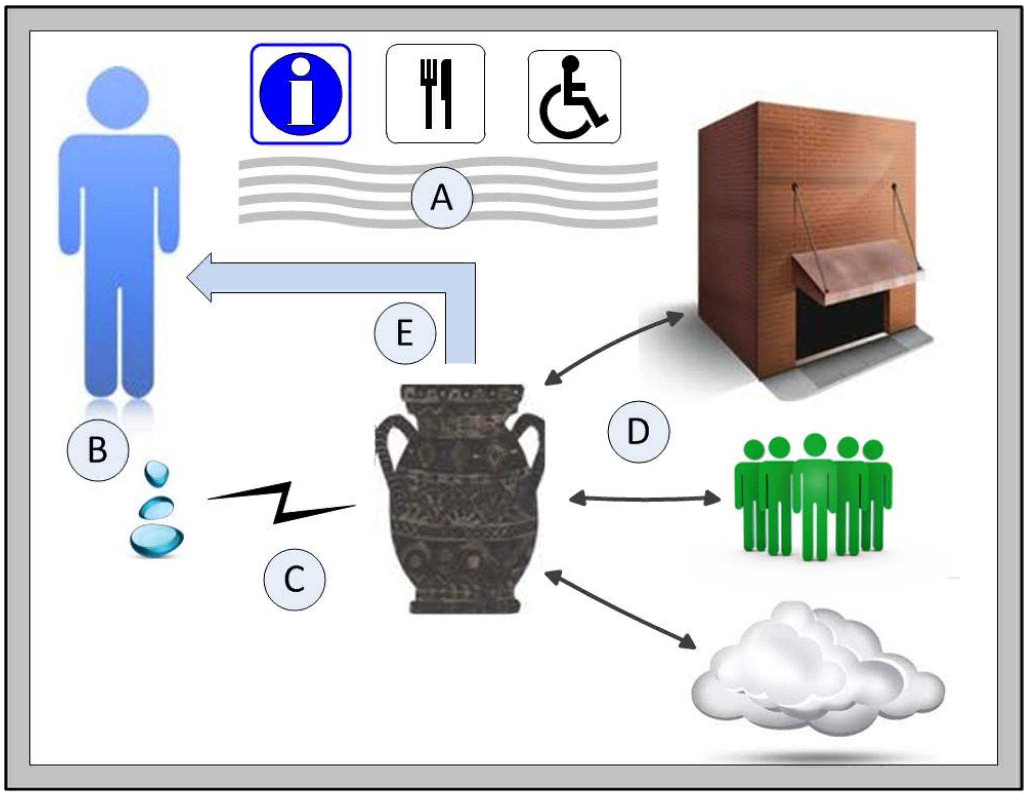


Figure 5 - Potential P&U Interactions

Possible P&U interactions are illustrated in Figure 5. P&U supports the situated interaction between a person and the places they encounter during their day-to-day activities (Interaction A in Figure 5). These activities may require an exchange of information about the place and related services. P&U can enhance the information exchanged between a user and a place and facilitate decision making. The relationship between the user and their Pebble (Interaction B) allows the Pebble to identify the user and provide content relating to current activities. Communication between Pebble and Urn (Interaction C) is a focal point of this research. This connection can vary based on specific P&U implementations or the connection characteristics of a particular user's Pebble.

Possible Pebble connection types include touching via near-field communication, such as using RFID or Bluetooth, scanning of tags like QR codes, or direct physical connection such as USB or Dallas Semiconductor's iButton. Rukzio and Broll refer to these connection types as physical mobile interactions or PMI [18]. An Urn could have several different physical interfaces to handle a heterogeneous mix of Pebbles. This concept encourages the investigation of a layered architecture where the physical communication layer uses "wrappers" to support different mechanisms for Pebble-Urn communication. The Urn can coordinate Pebble information with sources of place information (Interaction D). Opportunistic service identification, another research area of this proposal as identified by Research Question 1.2, occurs here. Information and services can be returned to the user (Interaction E).

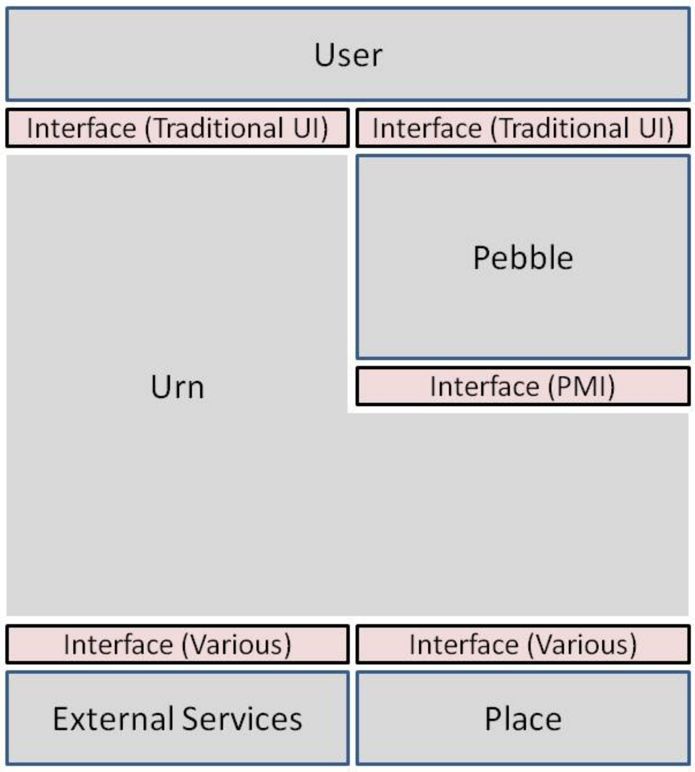


Figure 6 - P&U Interfaces

Figure 6 shows another representation of P&U interfaces. In addition to PMI connection types, the research focuses on novel ways Pebbles and Urns can manage communication sessions. For example, communication could be performed with a shared memory architecture using the Pebble Repository and Urn Repository. The other interfaces shown in Figure 6 represent traditional implementations and are not considered as part of the proposed research.

Figure 7 illustrates feature extensions of a P&U system. Shown here, the Urn controls actuators that alter the place environment, in this case lighting. The user's environment reacts to the presence of the user's Pebble. Lighting color within this place is controlled by a preference stored in the Pebble's user profile. In addition, the Urn alters the environment by coordinating the presence of multiple, different Pebbles.

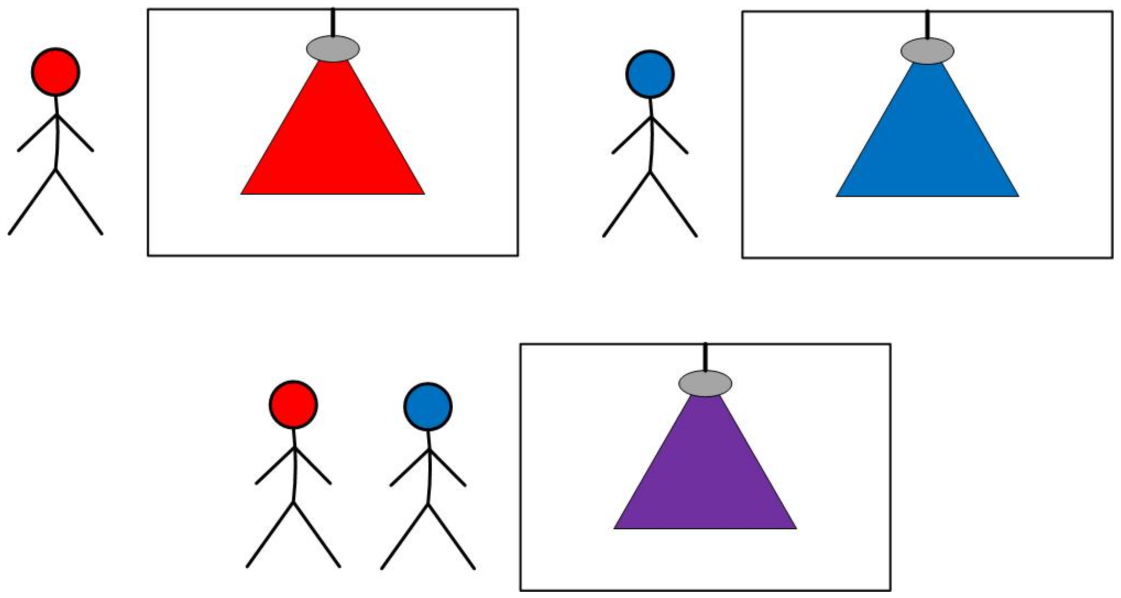


Figure 7 - Urn Coordination of Multiple Pebbles

2.4 Urn Services

P&U enhances the user's ability to accomplish common tasks in everyday life by augmenting their activities with information and services passed to them by a specific Urn. Urn information consists of user-generated content and context influenced by the Urn's place, environmental attributes provided by the Urn's sensors, content provided by users of the place, and, if supported by the local P&U infrastructure, information from a cloud of external services. This section describes aspects of P&U services.

P&U services are closely related to a specific physical location and to a particular situation of the user. Given these characteristics, P&U services may not be developed based on goal-directed activities [19]. Instead, P&U services possess a flexible composition formed by user and location contexts [20].

The need for several service discovery components is recognized in the P&U design. For example, service discovery scope, one of the challenges of opportunistic computing, is minimized by the P&U architecture due to the Urn being anchored to a physical location. The services provided by a particular Urn are bounded by the nature of its associated place. Similarly, services sought by a user interacting with a particular Urn are in a specific domain.

2.5 Pebbles and Urns Cyber Foraging

Cyber foraging is the opportunistic use of nearby computing resources (surrogates) by small, mobile devices [7]. Mobile devices achieve faster compute performance, access larger data stores, and conserve power by offloading tasks to more capable, nearby surrogate computers. Characteristics of the P&U system encourage and facilitate the use of cyber foraging strategies.

Section 2.2.1 describes Pebbles with varying capabilities based on their underlying implementation technologies. Pebbles that use microcontrollers and require power might take advantage of cyber foraging to reduce energy consumption when interacting with the Urn.

Section 2.2.2 discusses the Urn’s fixed location in a facility like a building or public area. This would insure to a large degree availability of continuous power and connectivity. Cyber foraging strategies could leverage these capabilities during Pebble and Urn interactions.

A network-attached Urn as discussed in Section 2.2.3 has access to external services such as remote execution and distributed storage. Strategies to move local responsibilities for these services away from the Urn could improve Urn implementation considerations of cost, performance or form factor.

2.6 Pebbles and Urns Use Cases

The following scenarios demonstrate ways in which P&U have the potential to enhance user experiences in common everyday activities.

2.6.1 Elderly and Assisted Living Care

Paul, a 78-year old man suffering from mild dementia, is intent on living alone and remaining self-supported as long as possible. This scenario describes how P&U assists Paul with remaining independent. Paul's dementia along with other medical conditions now prevent him from driving, He depends on public and other transportation services to attend to medical appointments, grocery shopping, the pharmacy, and other general travel activities. In addition, Paul's relatives live some distance from him, and are unable to visit him often.

Paul's P&U system is of a standard configuration. The Urn is a physical container with computing, network, and signaling capabilities. The Urn is located in the area of the house that Paul most frequently occupies. In Paul's case, the Urn is connected to a 24/7 "caregiver's" service that uses P&U to monitor Paul's presence in his house, manage schedules and appointments, assist in medication management, and determine travel routes. Paul's Pebble is a small physical artifact with data storage, biometric reader, and an Urn interface. Static information on the Pebble authenticates Paul's identity. Currently, Paul is at home and his Pebble is in the house Urn. The fact that the Urn contains the Pebble is of situational significance. Using established behavioral patterns, the caregiver service has some assurance that Paul is in his house because his Pebble is in the Urn. By accessing the caregiver service, loved ones can also see Paul is at home. In addition, when the Pebble is in the Urn, the Pebble's local storage is loaded with credentials and other information needed in conjunction with today's activities.

Paul has two appointments today. One is with a doctor's office across town, and the other is to refill medical prescriptions at the local pharmacy. Paul studies the appointment information on the Urn's visual display. As he removes the Pebble from the Urn and puts it in his pocket, a printed copy of his schedule is produced for him to take. According to the schedule, a driver from the caregiver service will arrive in five minutes to take Paul to his doctor's appointment.

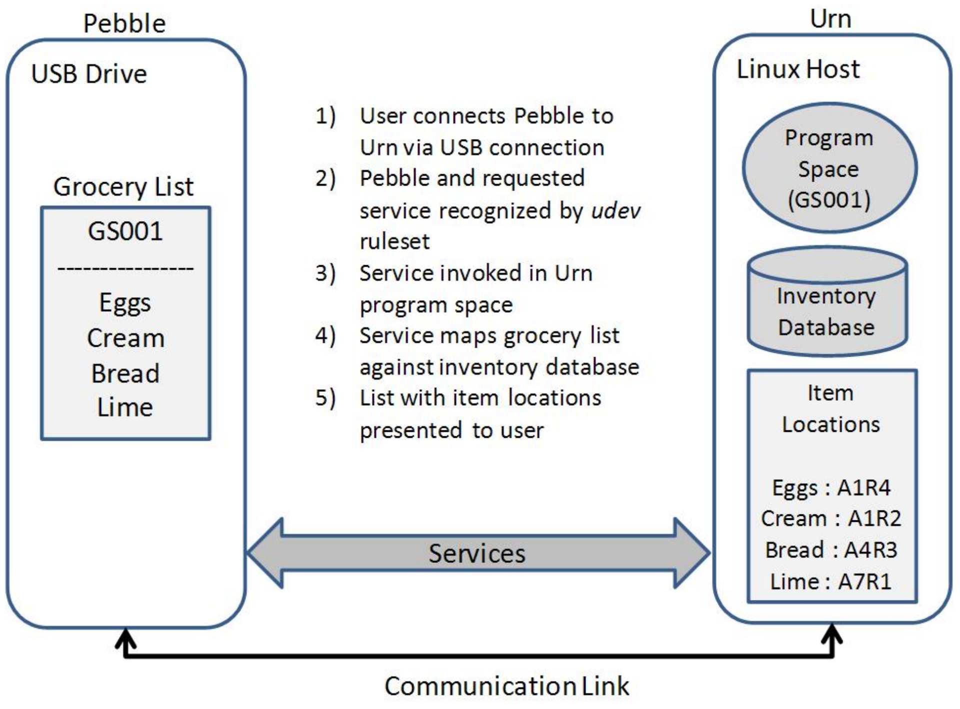
Stepping into the doctor's office, Paul scans his thumbprint with his Pebble, and places the Pebble in the office Urn. The office Urn accepts his identity, notifies the office system of his arrival, and returns his Pebble. The office system now retrieves Paul's medical records for the staff and the caregiver service is notified of Paul's arrival at the doctor's office. After his appointment, the caregiver driver returns to take Paul to his home.

That afternoon, Paul prepares to visit the pharmacy. Since the pharmacy is on the local bus route, Paul will take public transit. Removing his Pebble from the house Urn, he notices that the bus will arrive at his stop five minutes later than originally specified. At the pharmacy, Paul places his Pebble in the pharmacy Urn. Even though the pharmacy's network connection is down, Paul's prescriptions can be filled since his house Urn loaded his Pebble with current medication information and doctor certificates that morning.

Back home, Paul returns his Pebble to the house Urn, and promptly falls asleep in his recliner.

2.6.2 Grocery Store

Arriving at the grocery store, Pete deposits his Pebble containing his grocery list into the Urn located next to the carts. The Urn produces a printout, several coupons related to his grocery list, and returns his Pebble. Looking over the coupons, Pete certainly prefers receiving coupons prior to his checking out as opposed to afterwards. The printout contains his grocery list annotated with item and aisle locations. The current price of each item is also on the printout along with the total cost of the grocery list items.


Pebble
USB Drive
Grocery List
GS001

Eggs
Cream
Bread
Lime
Urn
Linux Host
Program Space (GS001)
Inventory Database
Item Locations
Eggs : A1R4
Cream : A1R2
Bread : A4R3
Lime : A7R1
1) User connects Pebble to Urn via USB connection
2) Pebble and requested service recognized by udev ruleset
3) Service invoked in Urn program space
4) Service maps grocery list against inventory database
5) List with item locations presented to user
Services
Communication Link

Figure 8 – Example P&U Grocery Store System

This location information speeds up shopping and the provided total cost of the shopping trip allows Pete to make any necessary budget decisions at the time. Pete notices from the printout that a couple of the items on his list are out-of-stock. The store system has recommended a replacement selection for one of the out-of-stock items. Figure 8 illustrates this simple P&U system.

2.6.3 Apparel Shopping

It is Saturday and not snowing; a good day for shopping. Donna begins her shopping day at Warmwater Brook, her favorite clothing store. The Warmwater Brook Urn is physically located in the store next to a large touch screen. Donna drops her Pebble into the Urn. Donna's Pebble contains the usual set of identifying credentials plus shopping information such as her clothing sizes and style preferences. As her Pebble connects to the Urn, the touch panel comes to life first welcoming Donna to the store. Next, the display contains a number of general store specials, including an unadvertised special of 25% off of all items in the store. The store Urn has access to inventory information not only for this store, but other area stores. Checking Donna's style preferences, in-stock items in her sizes appear on the touch screen. Donna navigates through the displayed inventory picking a style that she likes. Unfortunately, it is not the right color. However, the desired item in the correct size and color is in-stock at another store. Through the touch screen, Donna requests that this item be delivered to her local store that afternoon. Finished with her shopping, Donna retrieves her Pebble. The farewell message on the display informs Donna that she has just been issued a 10% discount coupon at Leaf and Nut that is valid for the next hour. Leaf and Nut is a restaurant near Donna's current location.

2.6.4 Freshman Orientation

Alice, an incoming freshman at Virginia Tech, walked out of the new Visitor's Center. She was excited that freshman orientation was finally here. Alice examined the object she had been given during check-in. The orientation staff had called it a Pebble, and described it as her key to information all over campus. The Pebble was a small cube with colored sides, except for one side that was marked with the text "QRP2751". The staff explained that she could use her Pebble to interact with Urns. Urns are located in every building on campus, and when activated by her Pebble would provide her with useful information about where she happened to be at the time. In most cases, Urns would be located inside the main entryway of each building. Most of the Urns looked the same; a Hokie Bird stature with camera lens for eyes and a large display on its chest. Alice was told that each color on her Pebble had meaning. If she held the blue side of her Pebble facing the Urn's camera, the Urn display would show her a room directory of the building she was standing in. If she used the red side of the Pebble, a map indicating locations of exits, stairways and restrooms would appear. Alice was told that the green side of her Pebble represented her identity. She could use this feature to mark her location on campus as she moved from Urn to Urn.

Continuing her orientation exploration, Alice headed for Squires Student Center. As she entered the building, she saw the Urn off to her left. Alice held the purple side of her Pebble next to the Urn's camera. This action displayed a list of current activities taking place in Squires. One activity starting in a few minutes was an introduction to the Credit Union. This is an area she wanted to know more about so she decided to go to this session. After the session, Alice decided to head back to her dorm. It was after 5:00pm so all dorms were now locked. A camera lens was mounted in the wall next to the entry door of her dorm. This camera was part of a Pebbles and Urns system responsible for access control to buildings. Alice showed the green side of her Pebble to the camera. The Urn knew that it was Alice's Dorm Urn and that it was being shown Alice's Pebble, so it unlocked the door for Alice.

2.6.5 Trip to the Arcade

Grabbing his Pebble from the family Urn, 12-year-old Kyle heads to the mall to meet his friends. He passes his Pebble though the Urn at the mall entrance, thinking this will make his mom happy. Upon sensing Kyle's Pebble, the mall Urn sent his mother a text message with a time, date, and entrance id stamp of Kyle's arrival. Kyle meets his friends at the FunTime Arcade. Kyle's and his friends' Pebbles contain a FunTime data extension. Stored in this data area are FunTime game statistics such as highest scores and most played games. Each time Kyle plays a game, he places his Pebble in an Urn receptacle on the game console. Game statistics are updated real time and scores authenticated against the player. Kyle's Pebble also contains FunTime approval information digitally signed by his mother, allowing him to play certain games with age limits or mature content.

2.6.6 Pebbles and Urns Deployment

Jonathan, a pervasive computing systems administrator (PC SysAdmin), is headed to his first job of the day, an installation of the latest P&U system in the local coffee shop. Jonathan is pleased this IoT phenomenon is catching on; business is crawling out of, or rather into, the woodwork. The version of the P&U system is one of the more popular configurations: a networked Urn with USB Pebble interface and standard sensor package. Arriving at the coffee shop, Jonathan saw that the baristi had set up a location for the Urn off to the side of the main entrance. Power and network attachments were easily accessible and Jonathan quickly had the Urn in place ready for initialization. Once activated, the Urn successfully went through its power-on and system test procedures. Jonathan plugged his administrative and diagnostic Pebble into the Urn. The Urn loaded its standard set of service routines from the Pebble. The

Urn first activated the network routines to check its Internet connection. Other routines registered this Urn with a P&U administration center. Others, using the Urn unique identifier, loaded external routines specialized to handle service requests that routinely occur in a coffee shop. Another routine tapped this Urn into the federation of Urns already up and running in this local geographical region. One of the final initialization steps involved this Urn connecting its local Urn Repository to the coffee shop's administrative data stores. Finally, the Urn transferred all installation logs back to the administrative and diagnostic Pebble. Jonathan removed his Pebble. The coffee shop Urn was up and running. Customers could now use this Urn to queue up and place orders and perform other P&U activities. Before leaving, he left 100 USB Pebbles with the baristi to be handed out to customers.

2.7 Summary

This chapter described conceptual aspects of the P&U architecture. The concept map in Figure 9 represents P&U components and their relationships.

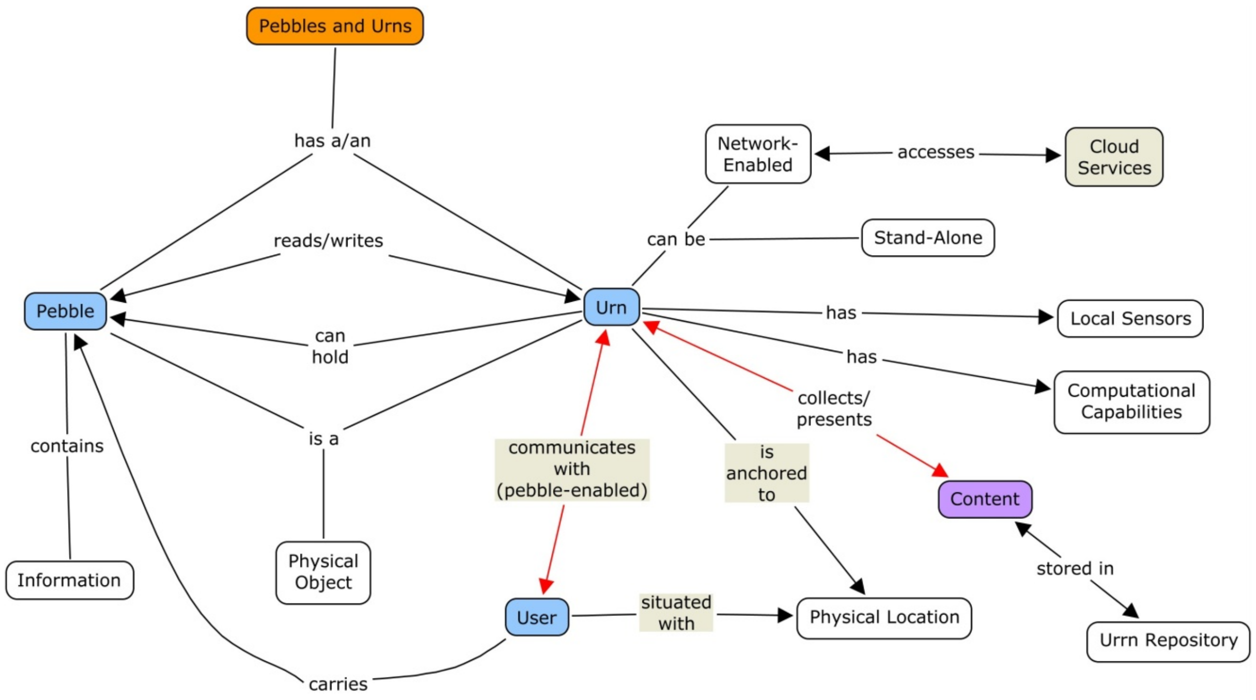


Figure 9 - P&U Concept Map

Key areas of the P&U architecture discussed in this chapter include enforced physical proximity of P&U interactions using PMI interfaces, Urn support of different Pebble classes, a layered P&U architecture, bounded context defined by the Urn's fixed location, and delivery of opportunistic, place-specific services

Chapter 3 provides technical background on the P&U concepts presented in this chapter.

Chapter 3: Overview of Interactions, Information, and Services

This research considers the proposed P&U architecture as a potential enabling technology for the design and implementation of the Internet of Things and opportunistic computing systems. The areas of pervasive computing and other fields contributing or relating to this architecture and its application cover vast bodies of prior and current research. This chapter provides necessary minimal background on the technical and conceptual areas germane to the P&U architecture and the related research questions presented in Section 1.2. Section 3.1 provides background and current work on mobile human-device and device-device interactions including tangible user interfaces, situated interaction and context, physical mobile interactions, place-specific computing, and opportunistic computing. Section 3.2 describes research areas contributing to the information management aspects of the P&U architecture. P&U information sources include place-generated and user-generated content. Pervasive computing service elements, including service discovery and cloud services, are discussed in Section 3.3. Cyber foraging is discussed in Section 3.4. Finally, Section 3.5 summarizes the distinctive characteristics of the P&U architecture when compared to current, related research activities.

3.1 Human and Device Interactions

The ongoing study of human-computer interaction (HCI) aggregates a collection of semi-distinct fields of research and practice in human-centered informatics. Although the P&U architecture contains HCI elements, the analysis of P&U from a HCI perspective is beyond the scope of this research and document. This section provides background on interaction concepts that influence the design of the P&U architecture. The research areas of tangible user interfaces, situated interaction, physical mobile interaction, place-specific computing, and opportunistic computing are represented in the pervasive computing timeline shown in Figure 10. A representative foundational paper for each research area is indicated in the figure.

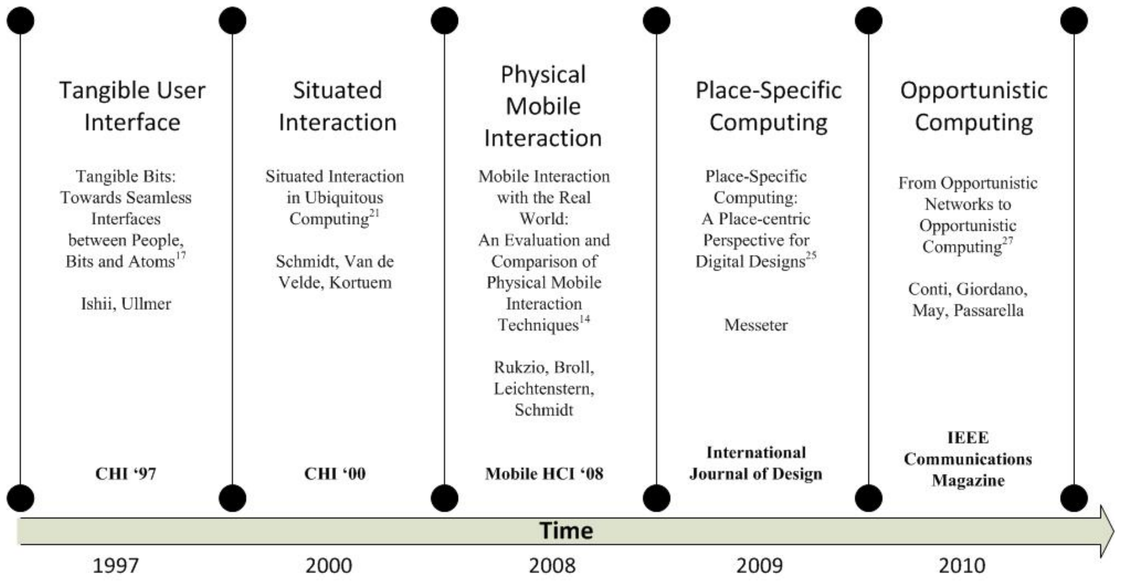

Tangible User Interface
Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms[17]
Ishii, Ullmer
CHI '97
Situated Interaction
Situated Interaction in Ubiquitous Computing[21]
Schmidt, Van de Velde, Kortuem
CHI '00
Physical Mobile Interaction
Mobile Interaction with the Real World: An Evaluation and Comparison of Physical Mobile Interaction Techniques[14]
Rukzio, Broll, Leichtenstern, Schmidt
Mobile HCI '08
Place-Specific Computing
Place-Specific Computing: A Place-centric Perspective for Digital Designs[25]
Messeter
International Journal of Design
Opportunistic Computing
From Opportunistic Networks to Opportunistic Computing[27]
Conti, Giordano, May, Passarella
IEEE Communications Magazine
Time
1997
2000
2008
2009
2010

Figure 10 - Interaction Research in Pervasive Computing

This timeline illustrates how areas of research in interaction have evolved into new areas of research. The HCI research that led to tangible user interfaces (TUI) moved computing from the desktop into our physical environment. Researchers began to explore the influences of situated awareness on computing interactions as they became part of a person's surrounding environment. Advances in the design and availability of mobile devices like personal digital assistants (PDA) and cellphones and the proliferation of wireless infrastructures offered new research opportunities to explore a person's computational interaction with other places, things, and people. Research in the field of interaction design, referred to as place-specific computing, stems from advances and previous studies in situated and context awareness, embodied interaction, and physical mobile interaction. Based on aspects of these research areas, people and places become physical components of a pervasive networking environment saturated with distributed resources. The studies of situation and context, mobile user interactions, place-specific computing, and emerging social networks lead to research on opportunistic computing where casual encounters between people and places offers opportunistic use of services and resources.

3.1.1 Tangible User Interfaces

Bishop's marble answering machine is one of the earliest illustrations of linking the physical and digital world [21]. The physical embodiment of phone messages as marbles illustrates a graspable physical container of information as part of a user interface. Tangible user interfaces are a collection of interfaces allowing users to interact collaboratively with augmented physical objects in order to access and manipulate digital information [22]. P&U is considered to have a TUI where the Pebble is a physical object containing information and the Urn is a physical container of pebbles.

3.1.2 Context and Situated Interaction

Dey and Abowd offer an application-oriented definition of context as [23]:

> Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.

Architect McCullough describes *situation* as when abilities connect with settings [24]. Situation defines how contexts shape perceptual selectivity, physical places focus social proceedings, and people act in their physical settings [24]. Situated interaction enables a computational system to support the users' activities in the context of use. Two classes of computational devices well-suited to use situational awareness are: 1) personal, mobile devices operated and operational while on the move, and 2) stationary devices in common spaces and changing environments [25].

German philosopher Heidegger proposes how computing artifacts can assist people in their daily activities and how they are situated in these activities [26]. Heidegger uses the term "available" to describe a relationship between a computing artifact, the user, and the task at hand. An "available" computing artifact allows the user to focus on what he or she wants to do. Heidegger believes there is a *future* aspect to people dealing with the present. According to Heidegger, human beings simultaneously: (a) exist in the world with particular interests (‘thrown’); (b) absorb in coping with the present (‘amidst’); and (c) press forward into future possibilities (‘projecting’) [26]. Given this orientation toward the future, a person’s everyday action in the world is guided by what he or she perceives to be the opportunities for action.

Opportunistic computing systems, described in Section 3.1.5, rely on the identification and use of correct contextual information to facilitate opportunistic encounters among people and places. Research Question 1.2 presented in Section 1.2 asks how the P&U architecture can assist in identifying the correct contextual information to use in a particular OC session. Two aspects of the P&U architecture supporting the use of situational awareness are: 1) Pebbles are personal devices operated by the user as they go about their daily activities, and 2) Urns are shared stationary devices associated with common places. Since the Urn is situated with a specific physical location, types of contextual information available in the Urn’s data repository are well-known prior to the occurrence of opportunistic encounters. This relationship contributes to research addressing Research Question 1.2 presented in Section 1.2. The Pebble can be considered as one of Heidegger’s “available” computing artifacts used to identify opportunities for action by the user.

3.1.3 Physical Mobile Interaction

Physical mobile interaction is an interaction paradigm that facilitates the use of mobile services through interaction with augmented physical objects. PMI uses mobile devices to extract information from augmented objects and to apply it for a more intuitive and convenient interaction with associated services [27]. Due to its increased simplicity and directness, physical interaction can make mobile interactions with “people, places, and things” more convenient and intuitive. This concept of the user perceiving the real world through device (computer) to device (computer) interactions is represented in Figure 11. This type of interaction occurs in the P&U architecture where the user’s Pebble (the mobile device) exchanges information with the Urn (the augmented physical object).

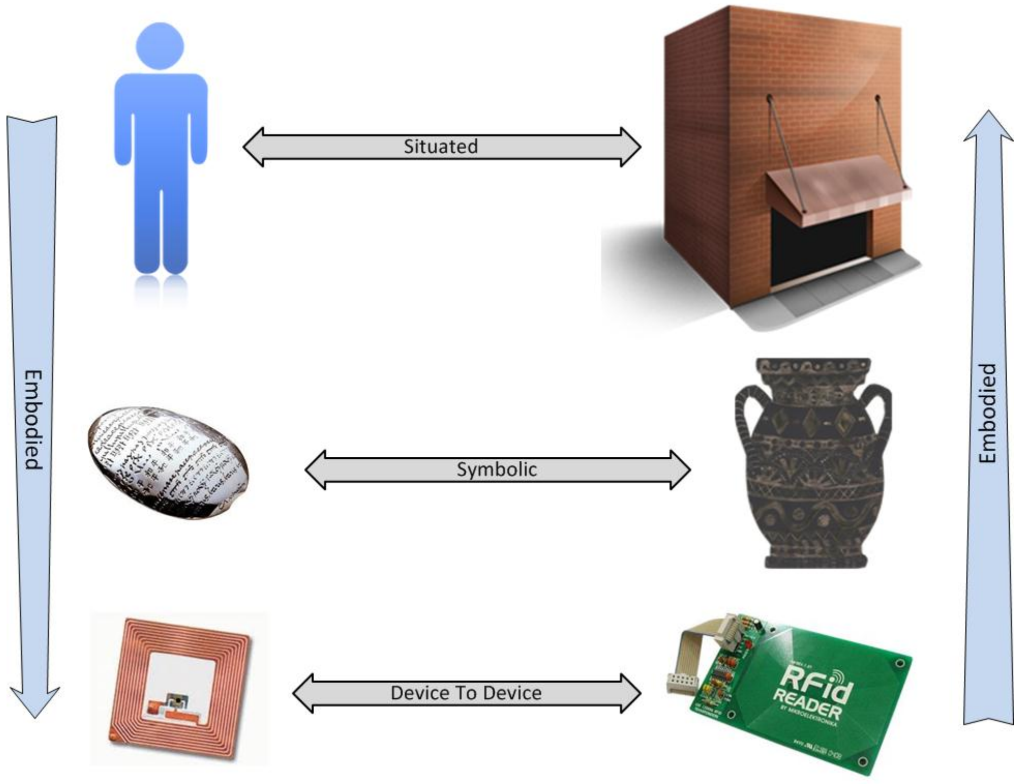


Figure 11 - Mobile Interaction with the Real World through Devices

PMI user interactions include touching, pointing, scanning, user-mediated data entry, and indirect remote control [18]. The touching interaction triggers an event when the user touches a smart object with a mobile device. Examples of technologies used to realize the touching interaction include near-field communication and proximity sensors. Pointing initiates an action when the user points at a smart object with a mobile device. Infra-red (IrDA) and visual marker (e.g., 2D code) technologies can be used to realize this interaction. The scanning interaction invokes an action when a user's mobile device is in proximity with a smart object. Bluetooth and other low-power radio frequency (RF) technologies can be used to implement this interaction type. With user-mediated interaction, the user communicates with the smart object via a mobile device text application. The Extensible Messaging and Presence Protocol (XMPP) could be used to implement this type of interface [28]. Indirect remote control involves the use of a mobile application to control a remote display.

In the P&U architecture, the Pebble is associated with an Urn in a physical manner. The physical device interactions described by PMI are ways that an association between a Pebble and an Urn can occur. The type of P&U interaction depends on the connection capabilities of a Pebble and an Urn. For example, a minimalist Pebble consisting of an RFID tag would be considered a PMI touching interaction. A Pebble with Bluetooth capabilities would be an example of a PMI scanning interaction. A Pebble-to-Urn USB interface would extend the PMI touching interaction.

3.1.4 Place-Specific Computing

Messeter defines place-specific computing as the shaping of interactions between people and place-specific resources, mediated by digital systems and services, and connected to global socio-technical networks [29]. He states that systems and services should be designed to cater to the people that, temporarily or regularly, visit or dwell in a particular place by providing cues for potential, or opportunistic, interactions. Systems designed in this way demonstrate a shift in mobile and ubiquitous computing perspectives from *anytime-anywhere* to *here-and-now*. Also, an Urn is a source of place-specific resources.

Seamon describes the importance of habitual movement in everyday life [30]. He classifies these movements into three categories: *body routine*, *time-space routine*, and *place ballet*. A *body routine* is a set of integrated actions that sustain a particular task like buying groceries. A *time-space routine* is a set of habitual bodily actions that extends for a considerable amount of time. Seamon defines a *place ballet* as an interaction of *body routines* and *time-space routines* rooted in a particular environment. When considering Research Question 1.2 presented in Section 1.2, future work would study the feasibility of incorporating Seamon's concept of a *place ballet* into the P&U architecture to identify relevant service domains and correct contextual information in support of an OC system.

The theory of ecological rationality accounts for how people make reasonable decisions given the constraints of limited time, information and computational resources that characterize most day-to-day experiences[31]. This research suggests that most decisions are made on the basis of short-cut strategies where people simply ignore most information available and focus only on the most useful and most prominent information [32]. Further examination of this research will address Research Question 1.2 and assist in the design of the P&U service composition component.

3.1.5 Opportunistic Computing

The emerging Internet of Things is producing a physical world saturated by fixed and portable devices with computing and communication capabilities. When two or more such devices come into contact, an opportunity occurs to share and exploit each other's software and hardware resources, exchange information, and execute tasks remotely. This interaction is called opportunistic computing [6].

OC systems include a social awareness component based on context management describing user behavior within social groups. Context representing human behavior can include, among other things, current user location, social links with other users, and users currently demonstrating similar social behaviors. This contextual information can be managed to provide cues for opportunistic interactions.

Anticipated challenges regarding the design and implementation of an OC system include [33]:

- the stability, duration, and frequency of a typical wireless connections among mobile users;
- availability and discovery of appropriate services and resources;
- incentives or reasons for initiating opportunistic conversations;
- implementation, management, and use of context describing user behaviors within social groups and physical places; and
- power management and other cyber foraging aspects of resource-constrained mobile devices.

The P&U architecture is proposed to help better understand and address the design and implementation challenges of an OC system. Research Questions 1.1, 1.2 and 1.3 listed in Section 1.2, address this objective.

3.2 Information Management and Sources

An integral part of the P&U architecture is the Urn Repository (UR). Information contained within the UR includes place information related to the Urn's location and user-generated content contributing to the location's social cache. Place information includes local content such as a restaurant's menu, inventory, and specials, and context generated by the Urn's sensor components like temperature, noise level, available seating, as well as location and current time. Program executables representing services tied to user activities are also stored in the UR. Contributing areas of research discussed in this section include database technologies, integration of sensor-generated information, and user-generated content.

3.2.1 Pervasive Computing Database Technologies

Considerations involving the design and implementation of an Urn Repository include decisions on the type of database to use, such as a traditional file system, tuple spaces, or an embedded database.

For a minimalist P&U architecture, the native file system of an operating system serves as the Urn Repository data store. As an example, guidelines on how to use the Linux file system as a database, including directory structure definitions and file naming conventions, are referenced in this article [34].

The Urn Repository of a more sophisticated P&U architecture involving federated Urns may best be implemented using tuple spaces. A tuple space is a model of distributed, associative memory within a concurrency framework [35]. Tuples are data elements in a tuple space. A tuple space provides asynchronous storage and retrieval of tuples, and access to tuples from any process on a distributed network. There are research activities that use the tuple space framework to implement resource discovery, context management, and other pervasive computing applications [35-37].

A P&U architecture supporting a large number of concurrent users, services and data transactions may require more sophisticated or specialized database technologies. A possible database candidate is a real-time, embedded database [38]. P&U is a data-driven system whose data resides in multiple locations and is accessible from multiple places involving connected or disconnected Urns. Embedded databases are designed to handle increased data sizes and the need for enhanced data processing capabilities required by mobile applications like P&U. Repositories of federated Urns using embedded databases would support transient data storage where data is streamed through the P&U applications in real-time and not permanently stored. Embedded databases are tightly coupled to the applications that use them. Data manipulation requirements of a P&U system are light to moderate, not complex in nature, and designed to perform a specific function. With a focus on the P&U application, everything is embedded in the application, including its database. Embedded databases are not separate from the application accessing them, and are deployed as part of that application installation. Embedded databases have the following characteristics [38].

1. Embeddable in applications
2. Small footprint
3. Run on mobile devices
4. Componentized data base management system (DBMS)
5. In-memory DBMS
6. Portable databases
7. Synchronize with back-end data sources

3.2.2 Sensor-generated Information

Data about the Urn’s associated physical place can be generated by sensors attached to the Urn or supplied by sensors distributed throughout the physical location. Implementations of sensor-based information systems vary in complexity. A small number of sensors directly attached to an Urn could be polled by the Urn Controller (UC) with resulting data stored in the

Urn Repository data store. More sophisticated distributed sensor arrays require enabling components that include a sensor database, sensor query structure, and a sensor query processing model. Cornell's COUGAR system is an example of a distributed sensor database system [39].

Message Query Telemetry Transport (MQTT) is an open-source publish/subscribe messaging protocol designed for resource-constrained devices and low-bandwidth, high-latency or unreliable networks [40, 41]. These characteristics make MQTT an ideal protocol to support connected devices in the IoT [42]. IBM Zurich Research Laboratory is developing an extension of the MQTT protocol called MQTT-S for use in wireless sensor networks (WSN) [43]. Protocol candidates for the P&U architecture are MQTT used in the Pebble-to-Urn communication layer, and MQTT-S used in the Urn-to-sensor communication layer.

3.2.3 User-generated Content

Researchers are studying ways in which user-generated content can improve understanding of user-place interactions and, thus, enable improved, customized services [44]. *Digital footprints*, produced by active and passive data collection during mobile device use, reveal individual and social behaviors of tourists with unprecedented detail [45].

Researchers at MIT are studying the phenomenon of *information scraps*, which are ubiquitous encodings of personal digital information generated by users during their day-to-day activities [46]. Examples of this loose form of information include electronic purchases, digital text files, self-addressed e-mails and electronic reminders. While this type of user-generated content might contribute to the support of opportunistic computing, its collection and use suggests a set of unmet design needs in current information systems. Design considerations of Urns and Urn Repositories in the P&U architecture offer ways to study how this information can be captured, used, and stored. For example, restaurant visitors would leave information like critiques of menu items and other content of potential value on the Urn location such as consumer patterns such as item preferences and queuing information.

Mamei and Zambonelli are studying a form of user-generated content in a distributed environmental memory based on the nature concepts of pheromones and stigmergy [47]. Users leave self-generated content as they interact with services within a physical location. Later visitors to this location encounter and interact with these information trails, and as stigmergy defines the actions of social insects, the visitors' local interactions are similarly mediated by the environment. Consideration of this work in the design of the P&U architecture addresses Research Question 1.2 presented in Section 1.2.

3.3 Services

A problem and solution interpretation of the P&U architecture is that the context of the user is the problem and the service is the solution. The support of opportunistic computing systems requires the P&U architecture be able to understand a user's activities and intentions and in turn provide the appropriate service or services. Service discovery, cloud services and cyber foraging relate to this need.

3.3.1 Service Discovery

More and more often, normal day-to-day activities include situations where embedded computational devices offer their services to the user. Mechanisms and processes are needed to support the identification of and interaction with services of most interest to the user in a given situation. Service discovery protocols facilitate interactions between heterogeneous devices with the aim of freeing users from tedious administrative and configuration work. Zhu has defined a service discovery protocol taxonomy that includes the following service discovery protocol components: service and attribute naming, initial communication methods, service discovery infrastructure, service information state, discovery scope, service selection, service invocation, service usage, and service status inquiry [48]. Zhu has defined three service discovery models [49].

1. Trivial – a client knows a service in advance, or the client has already cached the information about the service.
2. Client-Server – a client inquires about all services, and the server responds if a match occurs.

3. Client-Service-Directory – the client queries a directory of services, and then contacts the identified service of interest.

Kazhamiakin presents a model of service discovery based on the activities of the user [19]. Pervasive services accomplish tasks that are related with common tasks in the life of the user such as paying for parking or buying a bus ticket. These services are often closely related to a specific location and to the situation of the user. They are not characterized by a strong notion of a goal that must be achieved as part of a much broader plan, but they are used to address the contingent situation. This view presents a new way to look at service discovery that is centered on the activities of the user and the information that the user has available rather than the goals that a given service achieves.

A related area of research studies context-aware service discovery. With this type of service discovery, context is used to filter transmission of relevant information and services to the user. Current research by Rasch, *et al.* proposes a proactive service discovery approach named Hyperspace Analogue to Context (HAC) that continuously presents the most relevant services to the user in response to changes of context, services or user preferences [50]. Other research activities in this area of context-aware service discovery include [51-54].

As a proposed foundation for an OC system, the P&U architecture requires service discovery elements. As discussed, service discovery research indicates that only a small percentage of available services will be of interest to the user in any given situation. This condition is implicitly supported by the P&U architecture due to the situated location of the Urn relative to a physical place. Assuming a person is intentionally in a specific place for specific purposes, the Urn is able to offer only those services in support of those purposes. To support opportunistic sessions, the P&U architecture does depend on other service discovery features like service and attribute naming, initial communication methods, service information state, and service selection and invocation. The modeling and analysis of a P&U instance will help identify service discovery needs that could be implemented with existing or new protocols.

3.3.2 Cloud Services

One means of P&U service delivery is cloud computing. Cloud computing is a natural progression of service-oriented architecture. The definition of cloud computing used here is a computing paradigm that allows users to temporarily utilize computing infrastructure over the network, supplied as a service by the cloud-provider at possibly one or more levels of abstraction [55]. The levels of services described in this particular cloud computing ontology are listed below [55].

- Cloud Application – Software as a Service (SaaS)
- Cloud Software Environment – Platform as a Service (PaaS)
- Computational Resources – Infrastructure as a Service (IaaS)
- Data Storage – Data Storage as a Service (DaaS)
- Communications – Communication as a Service (CaaS)
- Firmware / Hardware – Hardware as a Service (HaaS)

Configured as a distributed model, the Urn Repository (UR) of the P&U system can be considered to be a cloud client [56]. A cloud client consists of computer hardware and software designed for access and delivery of cloud services. The UR would access the cloud DaaS if requested information is not locally available.

3.4 Cyber Foraging

Cyber foraging is the opportunistic use of nearby computing resources (surrogates) by small, mobile devices [7]. Mobile devices achieve faster compute performance, access larger data stores, and conserve power by offloading tasks to more capable, nearby surrogate computers. Task scheduling is an important component of any system using cyber foraging. Even in the simplest cyber foraging system, a task scheduler chooses where program execution occurs: the mobile device or the single, specific surrogate.

Kristensen, *et al.* describe a three-tier cyber foraging approach where a mobile device can choose between a local surrogate or remote cloud for resources [57]. There are three tiers of resources in this model: 1) the mobile device, 2) an opportunistically discovered surrogate, and 3) an Internet-connected cloud surrogate. A scheduling process checks appropriateness and availability of the resources in this order.

A different tiered approach to cyber foraging introduces the concept of cloudlets [58]. A cloudlet is a trusted, resource-rich surrogate computer that is well-connected to the Internet and is available for use by nearby mobile devices. The cloudlet infrastructure supports numerous decentralized cloudlets whose compute cycles and storage resources can be leveraged by nearby mobile computers. The cloudlet infrastructure demonstrates the use of cyber foraging techniques supporting mobile devices. Figure 12 shows cloudlet and P&U concepts.

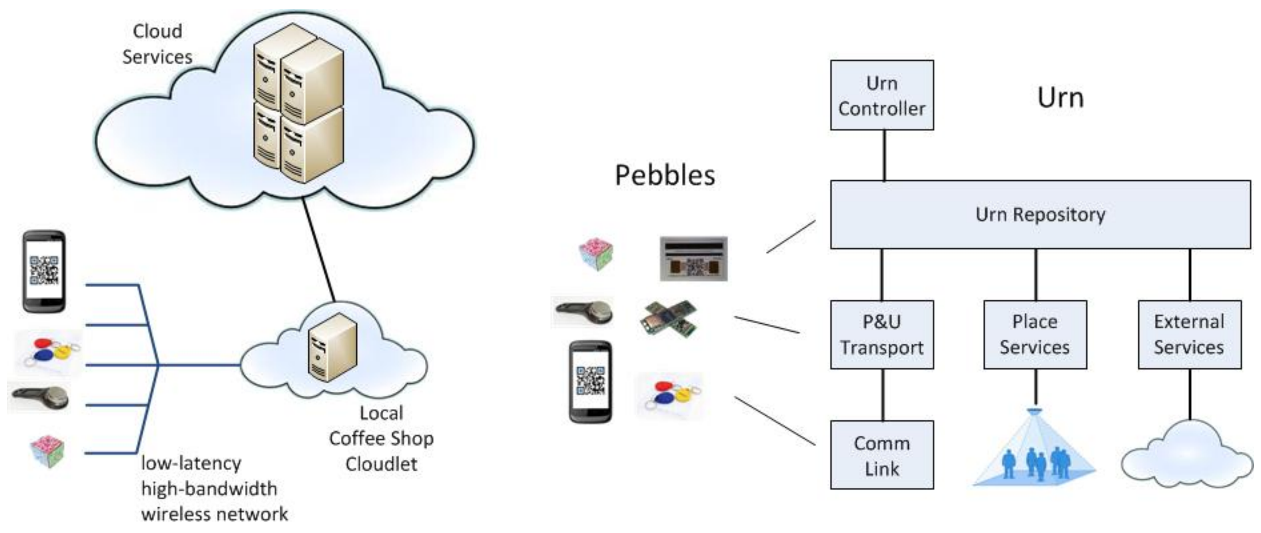


Figure 12 - Cloudlet and P&U Concepts

A number of cyber foraging frameworks have been developed. As an example, *Spectra* monitors the availability of resources and application resource usage in the local environment to decide when and where to use cyber foraging [59].

Research Question 1.3 presented in Section 1.2 specifically references cyber foraging techniques. Aspects of cyber foraging can be applied to the P&U architecture. The three-tiered cyber foraging model suggests a tiered structure for the Urn Repository. Figure 13 shows one possible configuration where a Pebble service request requires information not in the Urn Repository's local database.

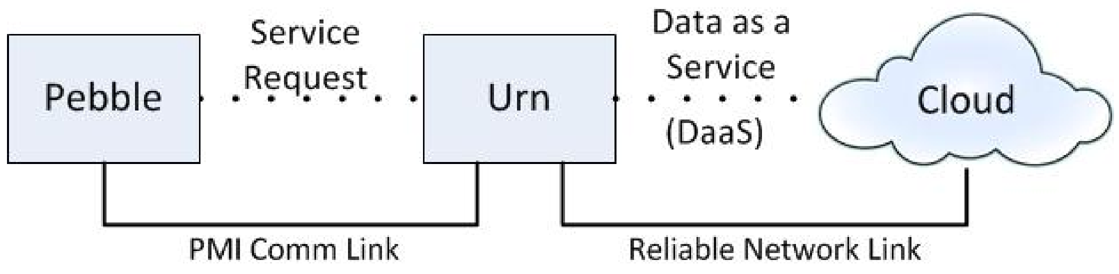


Figure 13 - Urn Repository Tiered Storage

Examination of the cloudlet infrastructure suggests ways in which P&U can leverage cyber foraging features. Cyber foraging monitor applications like *Spectra* demonstrate possible functionalities to include in the P&U architecture when multiple Pebbles of different classes interact with Urns in different physical locations.

3.5 Summary

This chapter provided minimal necessary overview and background on technical and conceptual areas relevant to the P&U architecture and the associated research questions presented in Section 1.2. Types of human-device and device-device interactions were introduced in Section 3.1. Interaction topics included tangible user interfaces, situated interactions, physical mobile interactions, place-specific computing, and opportunistic computing. Next, Section 3.2 described research areas contributing to information management of the P&U architecture. Current work in the areas of pervasive computing databases, user-generated content and sensor-generated data were briefly described. Finally, pervasive computing service elements, including service discovery, cloud services and cyber foraging, were discussed in Section 3.3.

Research areas discussed in this chapter were selected based on their relevance to the P&U architecture. The following paragraphs describe P&U architecture characteristics that share or complement characteristics and other aspects of these areas.

A piece of the P&U architecture's TUI is the Pebble carried by a person as they go about their daily activities. The Pebble can "know" the identity of its owner. The Urn is physically situated in the place that it embodies. The short-range characteristic of the PMI techniques used in the P&U architecture ensures the nearness of the user to the Urn and, consequently, to the place he or she is visiting.

Place-specific computing attempts to provide cues for opportunistic interactions during a person's activities within a specific location. Opportunistic computing is the interaction that occurs between two devices when they are able to share and exploit each other's resources and information. Urn characteristics, that include a fixed location, a capable computing platform, large amounts of storage including cloud access, a reliable network connection, and a continuous power supply, help alleviate several of the challenges associated with opportunistic computing described in Section 3.1.5.

Service discovery taxonomies include components to handle discovery scope. Recent and current research is investigating service discovery models that incorporate contextual information about user activities. The Urn knows who, what, and where it is with regards to location, time, Pebble owner, and a set of predictable activities. The activities are predictable in that, for example, a person shopping in a grocery store will not normally enter a waiting line to see a physician. The application of place-specific computing techniques using local sensor and user-generated information will refine the set of predictable activities within a place. Determining a set of predictable activities helps identify a relevant service domain and define correct contextual information used in an OC system; both contributing factors in addressing Research Question 1.2 presented in Section 1.2.

By the definition of cyber foraging systems, the P&U architecture with its small, mobile devices (Pebbles) and computationally-strong, situated surrogates (Urns) is a cyber foraging system. The study of service discovery models and cyber foraging systems provide insight into Research Questions 1.2 and 1.3 presented in Section 1.2 and assist with the P&U architecture interface design developed in Chapter 4.

The following chapter, Chapter 4, builds on the P&U conceptual framework presented in Chapter 2 by refining the P&U architecture to include service, information repository and application programming interface (API) definitions. Chapter 5 discusses prototypes of this model that are implemented and analyzed in conjunction with the research questions presented in Section 1.2.

Chapter 4: Pebbles and Urns Framework

Chapter 3 introduced technical and conceptual areas that influence and contribute to design of the P&U architecture. Topics including tangible interfaces, device-device interactions, place-specific computing, cyber foraging and service discovery were discussed to better understand P&U architectural features.

The design of the P&U architecture is influenced by multiple design factors, as introduced in Chapter 3. Each component of the architecture, including the Pebble, the Urn, each of their information repositories and the underlying communication infrastructure, has its own design requirements. This chapter examines the evolution of a P&U design model leading to the P&U framework. Section 4.1 examines design approaches and issues influenced by the research questions presented in Chapter 1 and the P&U conceptual model presented in Chapter 2. In Section 4.2, an initial design of the P&U architecture is studied. The evolved P&U framework, including descriptions of components, operational details, and API's, internal functions, and data structures, is presented in Section 4.3. Section 4.4 summarizes the design analysis of the P&U framework.

One area of work in this research consists of expanding the minimalist P&U design to include additional architectural components. Activities include continued evaluation of existing architectural styles and their application to P&U designs as well as development of new styles specifically designed to support a general P&U framework. Additional P&U instances based on these designs have been implemented and tested.

4.1 P&U Design Elements

This section discusses P&U architecture design paths and related issues derived from the research questions presented in Section 1.2 and a review of the P&U concepts and use cases presented in Chapter 2. Design analysis of these P&U concepts take into account the intended use of the P&U architecture to facilitate the implementation of opportunistic computing systems.

These derived attributes, discussed in the following sections, represent novel features of the P&U architecture.

4.1.1 P&U Interoperability

As presented in Section 1.1, interoperability is identified as one of the technical challenges associated with the underlying technologies of the IoT. Common practices and standards are required to support the diversity of the many different smart objects comprising the IoT. In the context of this research, interoperability is defined as the ability of an Urn to exchange information with different Pebble classes and to use the information that has been exchanged. P&U interoperability is substantiated by the extensible design of the framework's physical communication layer. Figure 14 shows the mapping of the research questions from Section 1.2 and the concepts discussed in Chapter 2 (with sections noted) into a list of P&U attributes related to interoperability.

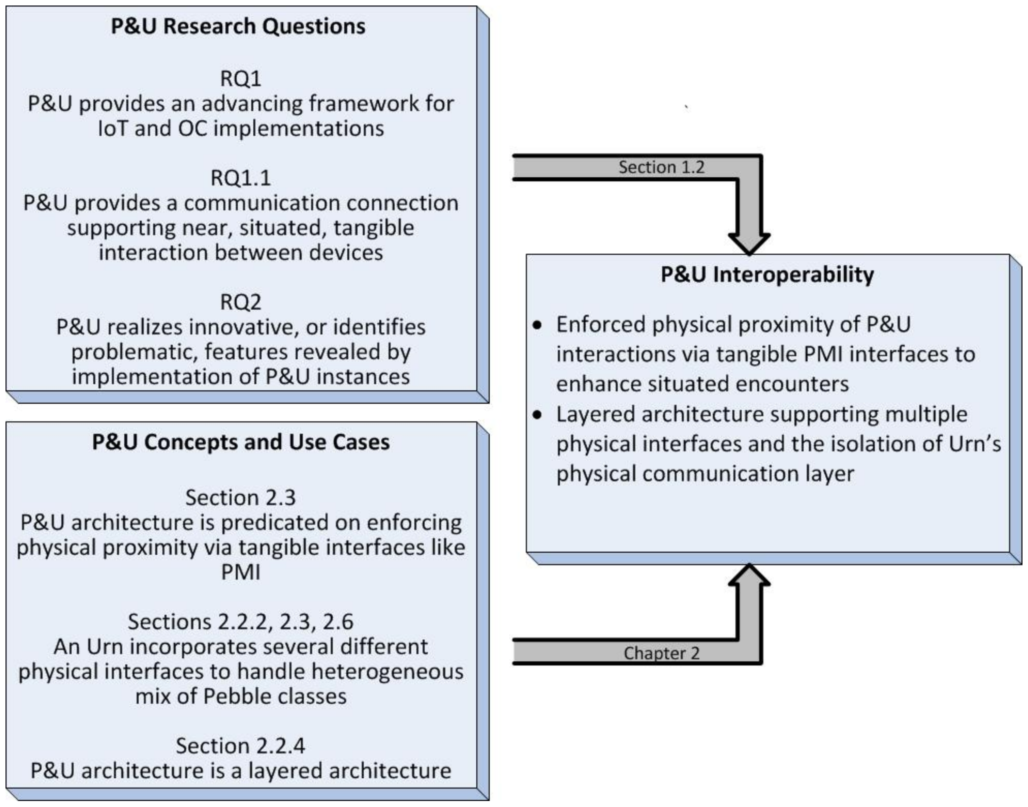

P&U Research Questions
RQ1
P&U provides an advancing framework for IoT and OC implementations
RQ1.1
P&U provides a communication connection supporting near, situated, tangible interaction between devices
RQ2
P&U realizes innovative, or identifies problematic, features revealed by implementation of P&U instances
P&U Concepts and Use Cases
Section 2.3
P&U architecture is predicated on enforcing physical proximity via tangible interfaces like PMI
Sections 2.2.2, 2.3, 2.6
An Urn incorporates several different physical interfaces to handle heterogeneous mix of Pebble classes
Section 2.2.4
P&U architecture is a layered architecture
Section 1.2
Chapter 2
P&U Interoperability
• Enforced physical proximity of P&U interactions via tangible PMI interfaces to enhance situated encounters
• Layered architecture supporting multiple physical interfaces and the isolation of Urn's physical communication layer

Figure 14 - P&U Interoperability Attributes

Research Question 1.1 and Section 2.3 identify a key feature of the P&U's interoperability design as the enforced physical proximity of Pebble and Urn interactions via tangible interfaces like Physical Mobile Interaction. The design requirement that P&U interactions occur in close proximity to each other results in situated interactions between the Pebble's owner and the Urn's place. Section 2.3 describes how the P&U infrastructure offers enhanced support of this situated interaction.

Another important interoperability feature established by Research Question 1.1 and Sections 2.2.2, 2.3 and 2.6 is the Urn's ability to interact with different Pebble classes using PMI interface technologies. Urn design includes specifications for multiple physical connections matching Pebbles based on RFID, QR code, Bluetooth or other mechanisms.

A supporting feature described in Section 2.2.4 is the specification for a layered P&U architecture that isolates the Urn from its physical layer of differing communication protocols and processes. A layered design also provides the opportunity to study P&U component interactions with a focus on identifying potential "thin waist" protocol implementations [60]. Section 2.2 presented the core components of the layered P&U architecture from a conceptual perspective.

Research Question 1 elicits research of these P&U interoperability features as advancing components of IoT and OC implementations. Research Question 2 addresses the innovative, or problematic, features encountered when implementing these features.

4.1.2 P&U Service Composition

P&U service composition involves the construction and delivery of an aggregate of opportunistic, place-specific services by the Urn. Figure 15 represents the mapping of the research questions and concepts from Chapter 2 into a list of P&U service composition attributes.

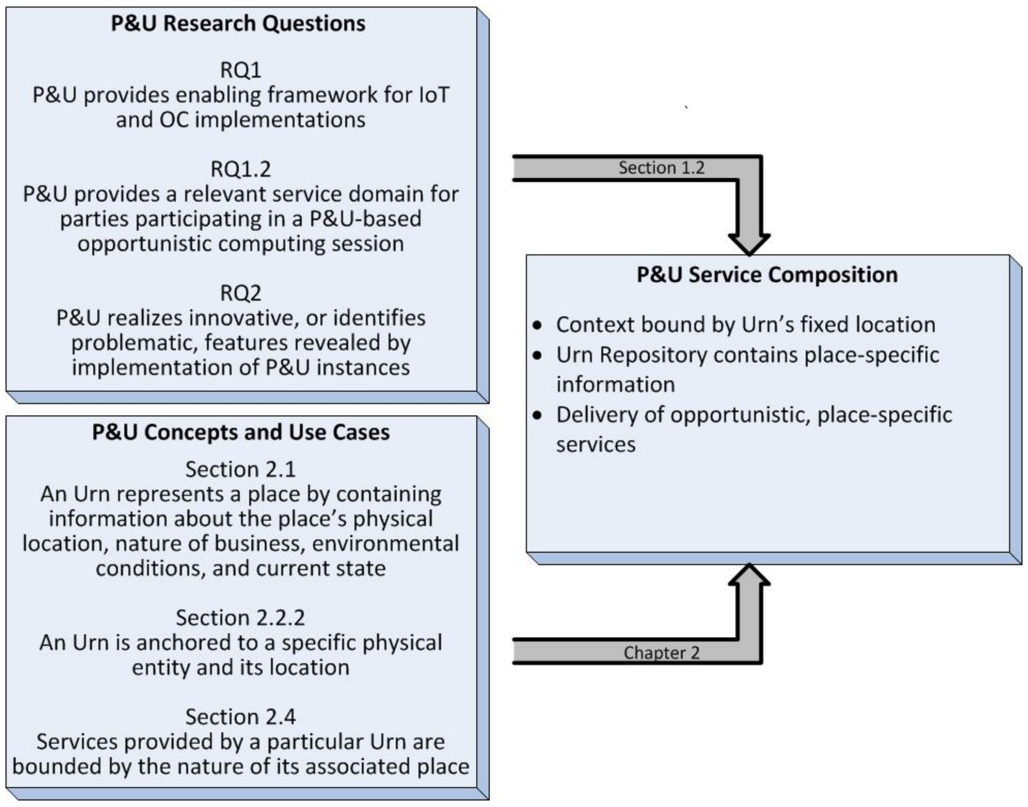

P&U Research Questions
RQ1
P&U provides enabling framework for IoT and OC implementations
RQ1.2
P&U provides a relevant service domain for parties participating in a P&U-based opportunistic computing session
RQ2
P&U realizes innovative, or identifies problematic, features revealed by implementation of P&U instances
P&U Concepts and Use Cases
Section 2.1
An Urn represents a place by containing information about the place's physical location, nature of business, environmental conditions, and current state
Section 2.2.2
An Urn is anchored to a specific physical entity and its location
Section 2.4
Services provided by a particular Urn are bounded by the nature of its associated place
Section 1.2
Chapter 2
P&U Service Composition
Context bound by Urn's fixed location
Urn Repository contains place-specific information
Delivery of opportunistic, place-specific services

Figure 15 - P&U Service Composition Attributes

The Urn's unchanging physical association with a place allows the Urn to represent that place as discussed in Section 2.1. The Urn's fixed location described in Section 2.2.2 is a key feature of P&U. The Urn's Repository contains information specific to its associated place such as the place's location coordinates, nature of business, offered services, customer activity and attributes such as current product inventories.

Section 2.4 extends this concept by explaining that the information and, therefore, services provided by a particular Urn are bounded by the nature of its associated place. This concept assists in addressing Research Question 1.2. Service requests and deliveries within the service domain of a P&U system can be bound by information contained in the Urn. A restaurant Urn need only provide information and services about the restaurant with which it is paired.

Research Question 1 considers P&U service composition as part of the proposed P&U framework. Research Question 2 identifies the innovative or problematic realizations made when implementing this service composition feature in a P&U instance.

4.1.3 P&U Cyber Foraging

Characteristics of the P&U system encourage and facilitate the use of cyber foraging strategies, described in Section 3.3.3, in areas of the P&U architecture. Figure 16 represents the mapping of the research questions and concepts discussed in Chapter 2 into a list of P&U cyber foraging attributes.

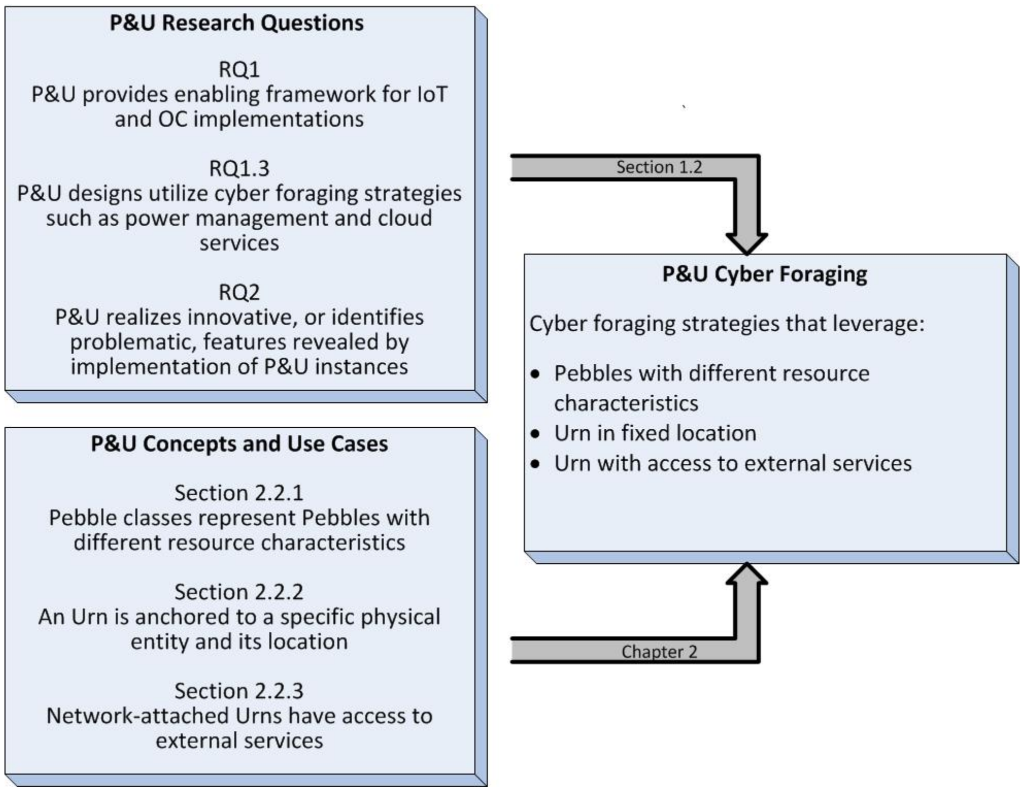

P&U Research Questions
RQ1
P&U provides enabling framework for IoT and OC implementations
RQ1.3
P&U designs utilize cyber foraging strategies such as power management and cloud services
RQ2
P&U realizes innovative, or identifies problematic, features revealed by implementation of P&U instances
P&U Concepts and Use Cases
Section 2.2.1
Pebble classes represent Pebbles with different resource characteristics
Section 2.2.2
An Urn is anchored to a specific physical entity and its location
Section 2.2.3
Network-attached Urns have access to external services
Section 1.2
Chapter 2
P&U Cyber Foraging
Cyber foraging strategies that leverage:
• Pebbles with different resource characteristics
• Urn in fixed location
• Urn with access to external services

Figure 16 - P&U Cyber Foraging Attributes

Section 2.2.1 describes classes of Pebbles with varying capabilities based on their underlying implementation technologies. Pebbles that use microcontrollers and require power might take advantage of cyber foraging to reduce energy consumption when interacting with the Urn.

Section 2.2.2 discusses the Urn's fixed location in a facility like a building or public area. This would insure to a large degree availability of continuous power and connectivity. Cyber foraging strategies could leverage these capabilities during Pebble and Urn interactions.

A network-attached Urn as discussed in Section 2.2.3 has access to external services such as remote execution and distributed storage. Strategies to move local responsibilities for these services away from the Urn could improve Urn implementation considerations of cost, performance or form factor.

Pebble classes and the Urn's available power and connectivity are P&U characteristics facilitating the study of Research Question 1.3. Research Question 1 considers the use of cyber foraging strategies as part of the proposed P&U framework. Research Question 2 identifies the innovative or problematic realizations made when implementing cyber foraging features in a P&U instance.

The attributes described in Sections 4.1.1, 4.1.2, and 4.1.3 aid in identifying the principal components, interactions, and configuration of the P&U system presented in Section 4.2.

4.2 Initial P&U Design and Framework

This section examines an initial design of a P&U general framework. Design considerations for this P&U architecture were obtained by applying the P&U requirements from Section 4.1 to the P&U diagram described in Section 2.2.4. The initial P&U design and framework are presented here as a foundation for the evolved P&U design and framework as discussed in Section 4.3.

4.2.1 P&U Early Architecture

The architectural design of P&U is based on the principal characteristics of proximity-based information delivery and simple, personal device interaction. Pebbles of different classes can possess different capabilities. This representation of the P&U architectural components and connectors considers a sophisticated "smart" Pebble with a UI, computational capabilities and read/write storage. The layered P&U architecture, where a function within one layer can obtain services from the layer below it, as illustrated in Figure 17, is comprised of two stacks of components.

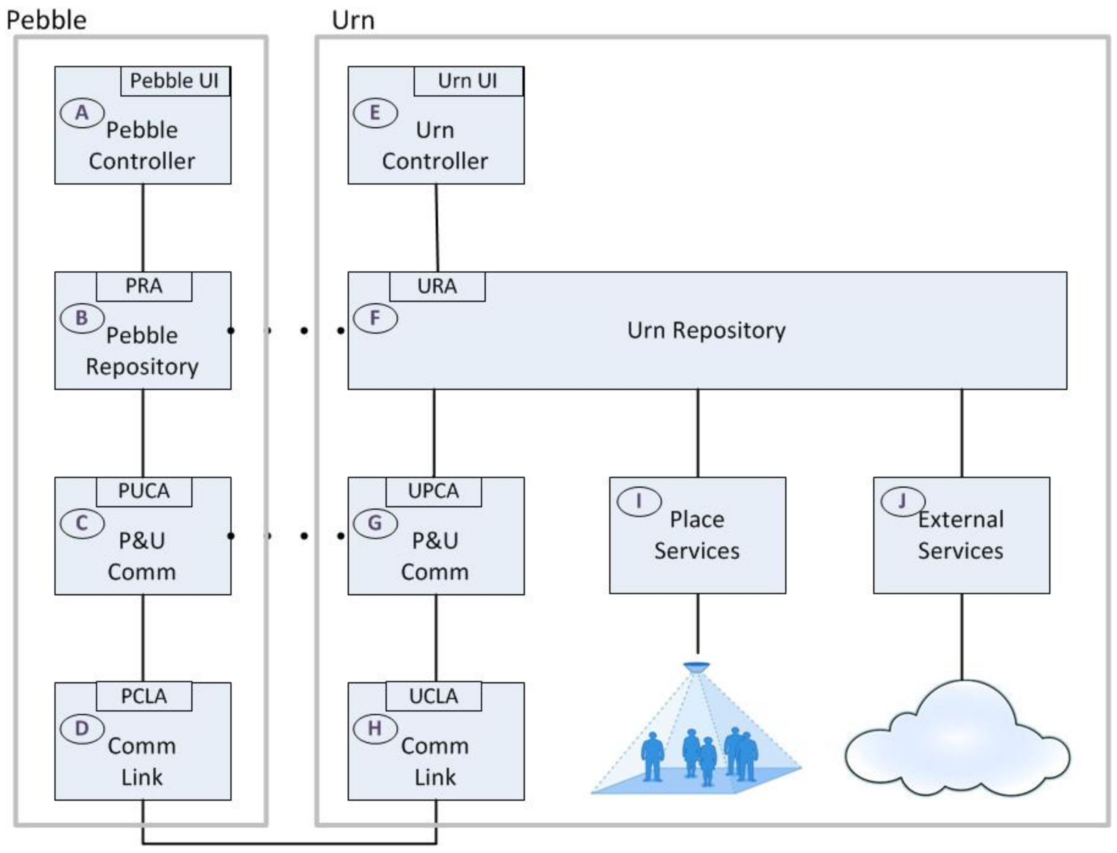


Figure 17 - P&U Components and Connectors

The Pebble stack contains the functional components of a Pebble: the Pebble Controller (A), the Pebble Repository (B), and Pebble communication to the Urn (C). In a similar composition, the Urn components are the Urn Controller (E), the Urn Repository (F), and the

Urn communication links to the Pebble (G). Place Services (I) support the interactions between the Urn and attached sensors that monitor the Urn's surrounding physical environment. External Services (J) consists of components and connectors that link the Urn to Internet-connected facilities or "cloud" services. The Urn interactions with Place and External Services are areas of future investigation and not discussed in this dissertation. The P&U communication protocol is represented by the dotted lines connecting pair (C, G). The P&U Repository protocol connects pair (B, F). The lowest layer protocol of the P&U architecture (D, H) is the physical communication mechanism that links a Pebble to an Urn. The layered design effectively isolates Urn components (G) and above from any coding or implementation changes in the physical communications layer. Layer (D, H) can use different networking and communication protocols and procedures to support the interoperability of different Pebble classes and Urn connectors. Possible Pebble connection types include touching via near-field communication (NFC), such as radio frequency identification (RFID) or Bluetooth, scanning of tags like quick response (QR) codes, or direct physical connection such as universal serial bus (USB). An Urn could have several different physical interfaces to handle a heterogeneous mix of Pebbles.

Communication with the Urn is facilitated via the message construct called the Pebble Message. Message fields include Pebble Identifier (PID), Pebble Class Identifier (CID), Pebble User Identifier (UID) and Urn Service Request (Svc_Req).

Section 4.3 contains a preliminary description of components and connectors that comprise the configuration of the P&U architecture. This architecture reflects the P&U attributes from Section 4.1 and aspects of architectural styles discussed in Section 4.2.

4.2.2 Initial P&U Component and Connector Design

This section describes the responsibilities and functions of the initial P&U architecture components and connectors (API's) depicted in Figure 17. The initial P&U architecture is incomplete in that some functionality is not described in detail or referenced. The evolved P&U architecture, described in Section 4.3, includes changes to several aspects of the initial architecture that follow from discoveries made during the research and addition design efforts.

A subset of the functions and data structures described in this section is used to implement and test a minimalist Pebble and Urn design in Chapter 5. Appendix A contains function parameters and data definitions for this subset.

4.2.2.1 Pebble

As discussed in Section 2.2.1, Pebbles of different classes possess different capabilities. This section considers a sophisticated Pebble with a UI, computational capabilities and read/write storage when describing Pebble components and connectors. Communication with the Urn is facilitated via the message construct called the Pebble Message.

4.2.2.1.1 Pebble Controller

The main responsibility of the Pebble Controller is to prepare the Pebble for presentation; to enable the Pebble to be detectable by an Urn. As the core component of the "smart" Pebble discussed further in Section 5.5 the Pebble Controller is also responsible for system initialization, Pebble UI management and communication with the Pebble Repository. Pebble Controller functions and their descriptions are contained in Table 3.

Table 3 - Pebble Controller Functions and APIs

Functions	Description
Pebble_Init ()	Performs initialization tasks on "smart" Pebbles Initializes Pebble UI Initializes Pebble Repository Sets Pebble state – state tables maintained in Pebble Repository
Pebble_UI ()	Manages "smart" Pebble user interface
Pebble_State ()	Monitors storage, memory and power on "smart" Pebbles

API	Description
PRA_Init ()	Initializes Pebble Repository
PRA_Data ()	Manages data for Pebble classes that support R/W storage
PRA_Urn ()	Defines contents of Pebble Message

The Pebble Controller communicates with the Pebble Repository through API functions PRA_Init(), PRA_Data(), and PRA_URN().

4.2.2.1.2 Pebble Repository

The Pebble Repository is the data storage component of the Pebble. The Repository consists of data management functions defined in Table 4 and data groups described in Figure 18. Responsibilities of the Pebble Repository include file input/output operations related to the Pebble UI and state management and the construction of the Pebble Message.

Table 4 - Pebble Repository Functions and APIs

Functions	Description
PRA_Init ()	Performs Pebble Repository initialization, open files
PRA_Data ()	Performs specified data operation on Pebble Repository
PRA_Urn ()	Constructs Pebble Message via PUCA API call

API	Description
PUCA_Present_Pebble ()	Additional preparation of Pebble Message – adds hash

The Pebble Repository communicates with the P&U Communications Link through the PUCS_Present_Pebble() API.

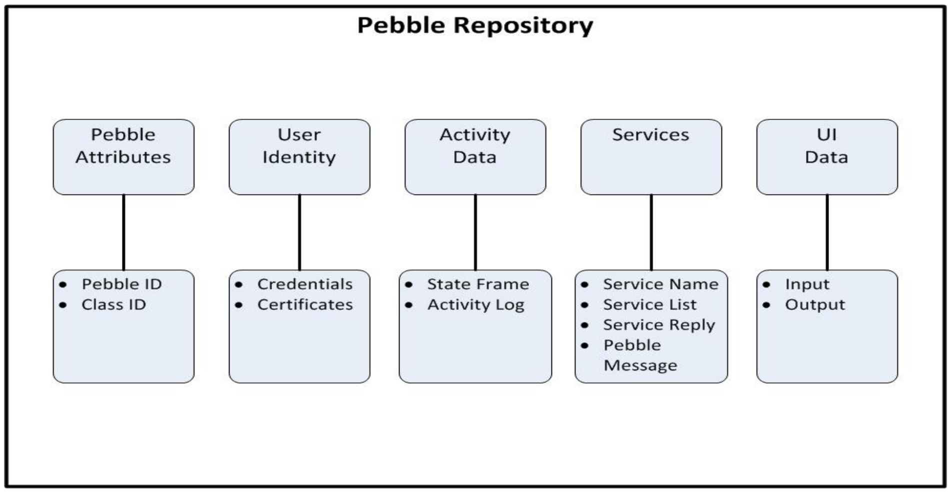


Figure 18 - Pebble Repository Data Groups

Table 5 contains descriptions of the Pebble Repository data groups.

Table 5 - Pebble Repository Data Group Descriptions

Data Groups	Description
Pebble Attributes	Files containing Pebble-specific information such as Pebble ID and Class ID
UI Data	Information displayed on or retrieved from the Pebble user interface
User Identity	Identity components of the Pebble owner used for authentication and authorization purposes
Activity Data	Tables containing information about state and session information and activity log
Services	Table containing Pebble service name and data and Pebble Message structure

Note that the Pebble Repository of the initial P&U design has limited functionality. Data groups are expressed in more detail for more capable Pebbles, as discussed in Section 4.3.

4.2.2.1.3 Pebble and Urn Communication Link

The P&U Communication Link component attaches a hash value of the constructed Pebble Message and passes the Pebble Message to the appropriate wrapper function for presentation to an Urn. Functions are described in Table 6.

Table 6 - Pebble Communication Link Functions and APIs

Functions	Description
PUCA_Present_Pebble ()	Generates hash of Pebble Message and sends to Comm Link for presentation
API	**Description**
PCLA_Urn_Comm ()	Invokes Comm wrapper

The P&U Comm Link invokes a communication wrapper function with the PCLA_Urn_Comm() API.

4.2.2.2 Urn

The Urn components and connectors described in this section mirror to a degree the Pebble elements of Section 4.3.1. As indicated in Section 4.3, the design of P&U's layered composition has evolved beyond this initial design.

4.2.2.2.1 Urn Controller

The Urn Controller is the core component of the Urn. It is responsible for processing the Pebble Message of a detected Pebble. Other Urn Controller operations include the following.

- Perform P&U system initialization
- Manage Urn UI
- Provide service delivery
- Process service requests
- Monitor for opportunistic conditions
- Communicate with Urn Repository via URA API
- Authenticate and authorize Pebble user
- Maintain P&U system state

These functions and APIs are described in Table 7.

Table 7 - Urn Controller Functions and APIs

Functions	Description
Urn_Init ()	Performs Urn initialization and startup • Initializes Urn UI • Initializes Urn Repository – URA_Init () • Sets Urn state
Urn_Shutdown ()	Performs orderly Urn shutdown
Urn_Restart ()	Performs Urn restart initiated by external trigger or error exception
Urn_UI ()	Controls Urn user interface
Urn_Opp ()	Determines additional service invocations given context of current Pebble-Urn interaction
Urn_Auth ()	Performs user authentication and authorizes access to services and information
Urn_Service ()	Handles service requests generated by Pebble contact • Invokes requested service • Accesses external and place services if needed • Sends service results to Urn UI • Updates Pebble Repository • Updates Urn logs and state tables

API	Description
URA_Init ()	Initializes Urn Repository
URA_Data ()	Data access to Urn Repository (logs, state tables, other data)
URA_Pebble()	This call blocks until a Pebble is ready to be processed

The Urn Controller accesses the Urn Repository by means of the URA_Init(), URA_Data() and URA_Pebble() function calls.

4.2.2.2.2 Urn Repository

The Urn Repository manages the information access and storage needs of the Urn. The Urn Repository is responsible for parsing an incoming Pebble Message into a data structure that is then passed up to the Urn Controller for further processing. Functions in the Repository maintain log and state tables with information about Pebble and Urn transactions. The Urn Repository honors information requests from the Urn Controller with access to Pebbles, external services and place-specific sensor data via function calls in their respective APIs. These function calls are UPCA_Get_Pebble(), ESCA_Ext_Svc() and PCA_Place(). Table 8 contains Urn Repository function descriptions.

Table 8 - Urn Repository Functions and APIs

Functions	Description
URA_Init ()	Performs Urn initialization • Initializes Urn Repository • Initializes Urn Comm Link
URA_Data ()	Performs specified data operation
URA_Pebble ()	Requests Pebble via UPCA API call – returns Pebble Message structure
UR_Process_Pebble()	Deconstructs Pebble Message, stores tokens in log file, loads Pebble Message data structure

API	Description
UPCA_Get_Pebble ()	Gets Pebble Message from active Pebble
ESCA_Ext_Svc ()	Accesses external services
PCA_Place ()	Accesses Urn sensors

Repository data groups are illustrated in Figure 19.

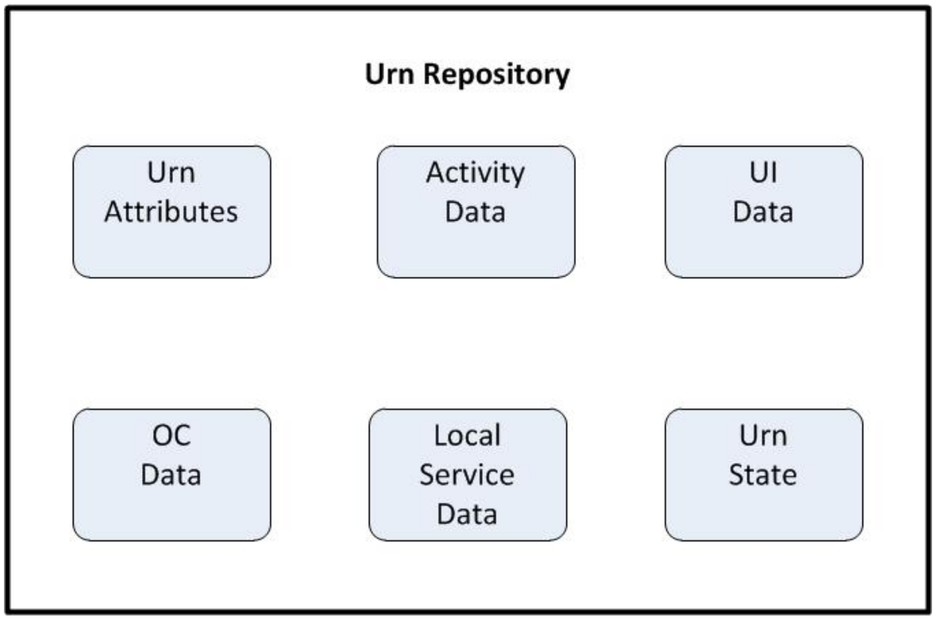


Figure 19 - Urn Repository Data Groups

Descriptions of the Urn Repository data groups are in Table 9.

Table 9 - Urn Repository Data Group Descriptions

Data Groups	Description
Urn Attributes	Tables containing information regarding a specific Urn (ex. Urn ID)
Activity Data	Tables containing information pertaining to Pebble and Urn interactions such as POI, Pebble encounters, place (sensors) and log files
UI Data	Tables containing information displayed on Urn user interface
OC Data	Tables containing rulesets used in determining opportunistic situations
Local Service Data	Tables containing information regarding services available from a particular (this) Urn
Urn State	Table containing Urn state information such as Urn status and current Pebble interactions

An anticipated outcome of this research is the refinement of the Urn Repository data groups as progress on implementation and testing of P&U design progresses.

4.2.2.2.3 Pebble and Urn Communication Link

The P&U Communication Link verifies the hash of the Pebble Message obtained from the Pebble wrapper function. Function descriptions are in Table 10.

Table 10 - Urn Communication Link Functions

Functions	Description
UPCA_Get_Pebble ()	Invokes specific Pebble Comm wrapper via UCLA_Wrapper () API call. Performs hash check on returned Pebble Message.
ESCA_Ext_Svc ()	Accesses external services
PCA_Place ()	Accesses place sensors

Table 11 describes the Urn Communication Link APIs'

Table 11 - Urn Communication Link APIs

API	Description
UCLA_Wrapper()	Invokes one of following functions: • QR_Code_Reader () • RFID_Tag_reader () • USB_Device_Driver () • Hybrid_Pebble_Reader () • Smart_Pebble_Reader () • Urn_Sensors () • Cloud_Access ()

The UCLA_Wrapper() API isolates the above Urn layers from different physical communication links. The initial P&U architecture is demonstrated using QR codes as the PMI method. Additional PMI techniques are utilized in the evolved P&U including USB and "Smart" Pebble interfaces, in addition to QR codes.

4.2.2.3 Pebble and Urn Messages

This section describes an initial Pebble Message format presented in Figure 20. More involved message formats are likely to be developed during the course of this research.

PID	CID	UID
Svc_Req		Msg_Hash

Figure 20 - Simple Pebble Message

Message fields are as follows.

- PID – Pebble Identifier
- CID – Pebble Class Identifier
- UID – Pebble User Identifier
- Svc_Req – Urn Service Request
- Msg_Hash – Computed hash of the Pebble Message

4.2.3 Evolution of Pebbles and Urns

This section describes the evolutionary factors contributing to the transformation of the initial P&U design presented in Section 4.2 into the P&U design and framework presented in Section 4.3.

The Pebble of Interest (POI) application described in Section 5.1 is a simple P&U instance built on the initial P&U design presented in Section 4.2.2. An examination of this prototype with respect to the research questions discussed in Section 1.2 reveals several open issues with the original P&U design. To further address and answer the research questions, a more complete P&U model and collection of derived prototypes are required.

Research Question 1.2 focuses on ways in which a P&U system might be able to deliver opportunistic services and information. Combining the research motivation provided by Research Question 1.2 with a review of current work in the areas of service composition, delivery, and knowledge management identified the need to enhance the initial P&U model's management and composition of Repository content and context. An example of current work in knowledge discovery involves studying the effects of user profiles and preferences on the accuracy of selection in online dating services [61]. This and other work encouraged the exploration and development of Repository profile and preference data structures. Other Repository enhancements include the addition of data elements supporting a simple service request process and a caching mechanism that collects P&U session information. These additional P&U Repository data structures and associated data manipulation routines aid in better understanding RQ 1.2 by defining elements of a P&U service domain that assist with the identification of opportunistic situations. Sections 4.3.2.2.1 and 4.3.2.3.2 provide detailed discussions of these data structures and routines.

The layered architecture of the initial P&U model served as a starting point in addressing RQ 1.1. To further study the flexible Pebble and Urn interactions noted in RQ 1.1, the isolated communication layer in this architecture is now extended to support different Pebble classes.

Research Question 1 emphasizes investigating ways to make P&U suitable as an Internet of Things deployment framework. Study in this area has resulted in the further design and development of APIs and toolsets that can be used by P&U developers to create various and unique P&U applications supporting different Urn locations and Pebble classes.

With respect to RQ 1.3, the ability of the P&U framework to support different Pebble classes allows the exploration of ways in which cyber foraging techniques might apply to P&U. Different Pebble classes and Urn interactions also offer the opportunity to obtain performance metrics within areas of the P&U framework.

Experience gained by developing P&U prototypes incorporating different Pebble classes led to insights into user privacy and roles. Developing controls afforded by lessons learned during Read/Write and Smart Pebble design efforts resulted in the P&U user having the ability to actively select facets of personal information used to represent them during a P&U session.

Section 4.3 presents an evolved P&U model with the following features.

- Distinct, well-defined architecture layers
- Expanded APIs and toolsets supporting P&U application development
- Design of additional Pebble classes and instance implementations
- Detailed Repository data structures providing contextual information in support of opportunistic service delivery
- Added controls for user management of privacy and roles

4.3 Pebbles and Urns Framework Design

This section describes the P&U framework that is based on research guided by the initial P&U design described in Section 4.2 and activity to address and answer the research questions presented in Chapter 1.

4.3.1 P&U Architectural and Operational Overview

The P&U architecture state diagram is modeled after a network protocol state diagram, specifically the Transmission Control Protocol (TCP) [62]. Figure 21 shows the state diagram for TCP which can be compared to the state model developed in this research for P&U.

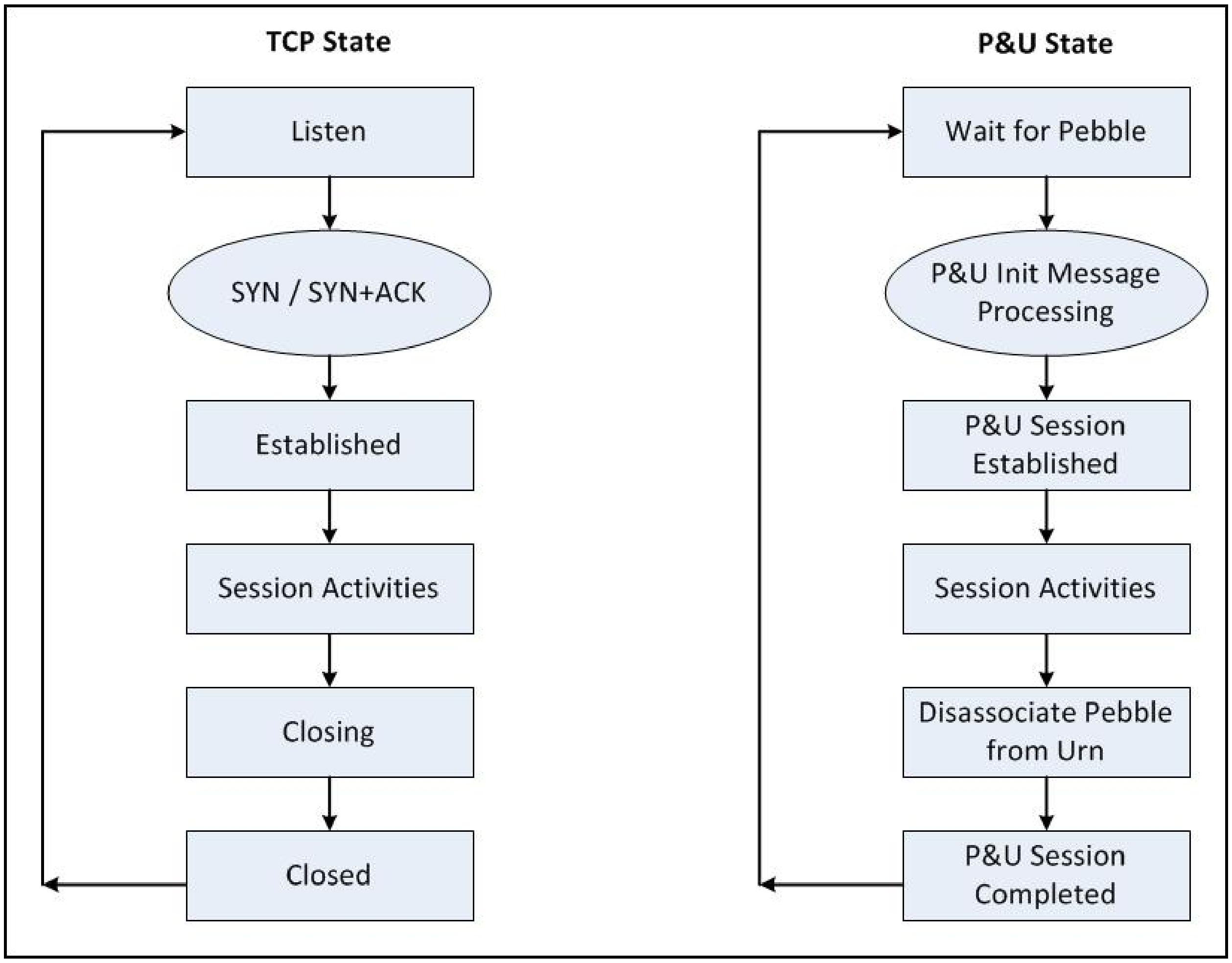


Figure 21 - TCP and P&U State Diagrams

The initial state of the P&U system is the Urn, in its fixed location, waiting for the presence of a Pebble. When a Pebble is detected, a Pebble initialization message (PIM) and an Urn initialization message (UIM) are processed by the Urn Controller. Next, the P&U session is established. Pebble and Urn Repository elements are enabled and made available to the Urn

Controller. In the next phase, the Urn Controller executes P&U session activities. These activities include service request and delivery, profile and preference comparisons, and previous session log traversals. At the end of session activity, the Pebble and Urn Repository elements are updated. Next, the P&U session completes and Urn control returns to the wait-for-Pebble state.

4.3.2 P&U Framework Layers

Due to the evolution and maturity of this P&U design, the following P&U framework representation emphasizes the architectural layers instead of the stack description used in Section 4.2.2. Two versions of the P&U framework are presented in Figure 22.

Figure 22(a) represents the architecture layers and components that support Read-Only and Read/Write Pebble classes. These Pebble classes do not have a Pebble Controller component. These Pebble classes only provide or provide and store information for the Read-Only or Read/Write Pebble class, respectively. These two Pebble classes also do not actively control communication with the Urn, so there is, also, no P&U Transport component. Figure 22(b) shows the layers and components that support the Smart Pebble class. The Pebble Controller component allows a Smart Pebble to process and control information, as well as provide and store information. A Smart Pebble can also actively control the communication through the P&U Transport component.

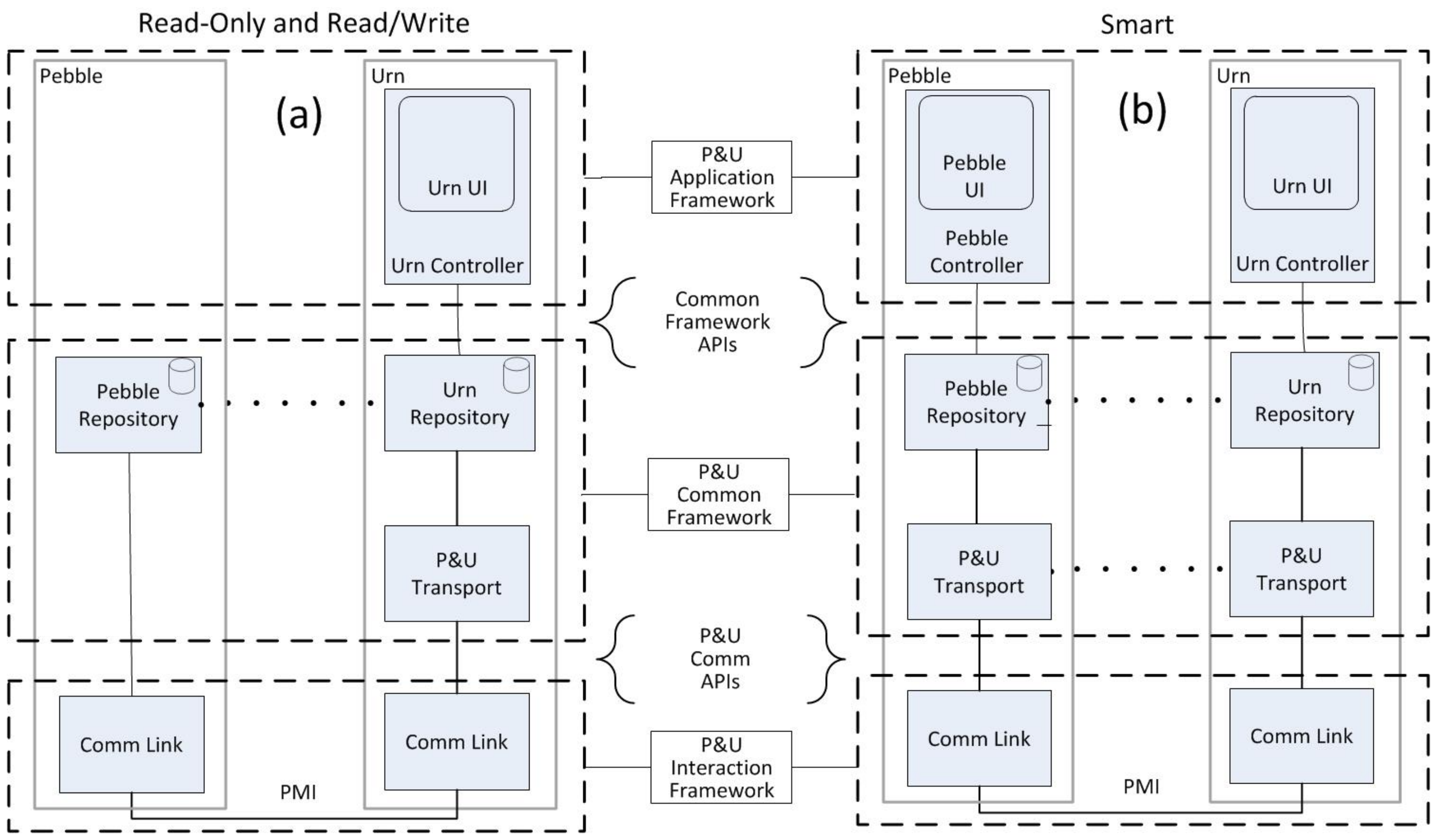


Figure 22 - P&U Framework

The following sections describe each layer and its components, programming interface, and internal functions.

4.3.2.1 P&U Interaction Framework

The P&U Interaction Framework manages the physical communication link between Pebble and Urn. This framework layer is the isolated physical layer identified as a P&U design element in Section 4.1.1. This framework supports Urn interaction with the three Pebble classes described in Section 2.2.1 by providing communication wrapper services to the P&U Common Framework layer. The communication wrapper for each Pebble class uses a physical interaction technology that supports the P&U interaction characteristics described in Research Question 1.1. New Pebbles and Pebble classes can be supported by writing additional wrappers. Table 12 shows the interaction technology used by each Pebble class.

Table 12 - Physical Interaction Technologies of different Pebble classes

Pebble Class	Physical Interaction Technology
Read-Only	QR Code Scan
Read-Write	USB / udev
Smart	Bluetooth

P&U Interaction Framework functions are described in Table 13.

Table 13 – Interaction Framework Functions

Urn Functions	Description
WRAP_Comm()	Provides Pebble Repository access to the Urn Common Framework by means of the following wrapper functions
Get_QR()	Transfers text from Read-Only QR Pebble
Link_USB()	Links the Read/Write Pebble Repository to the Urn Repository
Transfer_BT()	Transfers Smart Pebble Repository to and from the Urn Repository
Smart Pebble Functions	**Description**
Transfer_BT()	Transfers Smart Pebble Repository to and from the Urn Repository

Appendix B.1 contains function parameters and data definitions for this subset.

4.3.2.2 P&U Common Framework

The P&U Common Framework supports P&U system development and deployment by connecting the P&U application environment to the P&U physical interaction link by means of application and physical interaction API's and consistent data repositories. The following sections describe the P&U Common Framework data repository structures and the P&U Transport component.

4.3.2.2.1 P&U Repository Structures

Pebble Repository data elements are shown in Figure 23. Repositories of different Pebble classes contain all or a subset of these data elements based on their storage and data access capabilities.

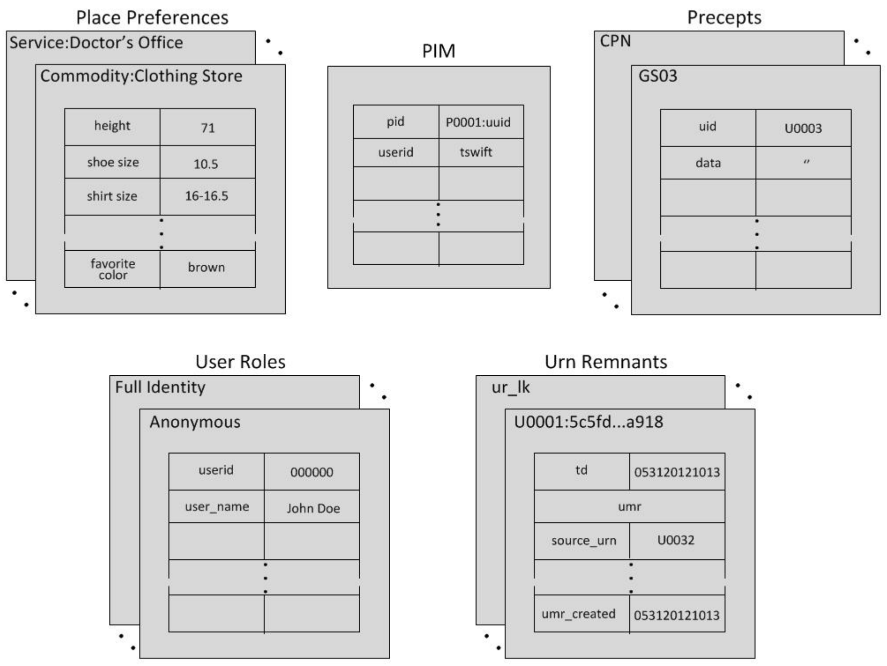


Figure 23 - Data Elements of Pebble Repository

Descriptions of the Pebble Repository data elements are contained in Table 14 and the glossary in Appendix E. Appendix C.1 contains the data dictionary for the Pebble Repository data elements.

Table 14 - Pebble Repository Data Element Description

Data Elements	Description
Place Preferences	User preferences relative to services provided by a place
PIM	Pebble Initialization Message containing Pebble ID (pid) and Pebble class interface (cid)
Precepts	Service requests
User Roles	Pebble owner profiles (Smart Pebble)
Urn Remnants	Snapshot data representing previous Urn sessions involving this Pebble

Figure 24 shows the data elements comprising the Urn Repository. This repository configuration is consistent across all Urn's with respect to naming conventions, data types and data formats.

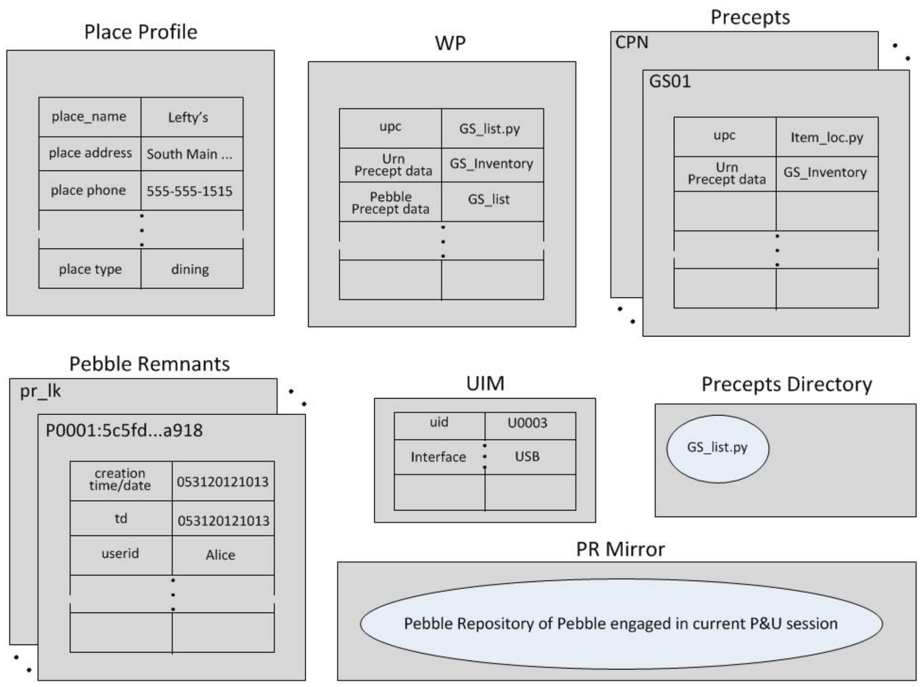


Figure 24 - Data Elements of Urn Repository

Table 15 contains descriptions of the Urn Repository data elements.

Table 15 - Urn Repository Data Element Description

Data Elements	Description
Place Profile	Profile information of the place represented by the Urn
Working Precept (WP)	Data mashup of Pebble and Urn precept elements supporting a successful service delivery/request process
Precepts	Services offered by this Urn
Pebble Remnants	Snapshot data representing previous Pebble sessions involving this Urn
UIM	Urn Initialization Message containing Urn ID (uid) and Urn physical interface
Precepts Directory	Urn Repository area containing service executables and data
PR Mirror	Mirrored Pebble Repository during active P&U session

A data dictionary of the Urn Repository data elements is contained in Appendix C.4.

4.3.2.2.2 P&U Transport

The P&U Transport component is responsible for providing Pebble Repository access to the P&U Application Framework. P&U Transport accomplishes this task by replicating, or mirroring, the Pebble Repository onto an Urn Repository data element. Specific implementations of the access process vary based on Pebble class. The details of the different access implementations are hidden from the P&U developer. At the end of the P&U session, the Pebble Repository is reloaded with updated data elements from the P&U session. Figure 25 illustrates the Pebble Repository replication process. The WRAP_Comm functions are described in Table 13.

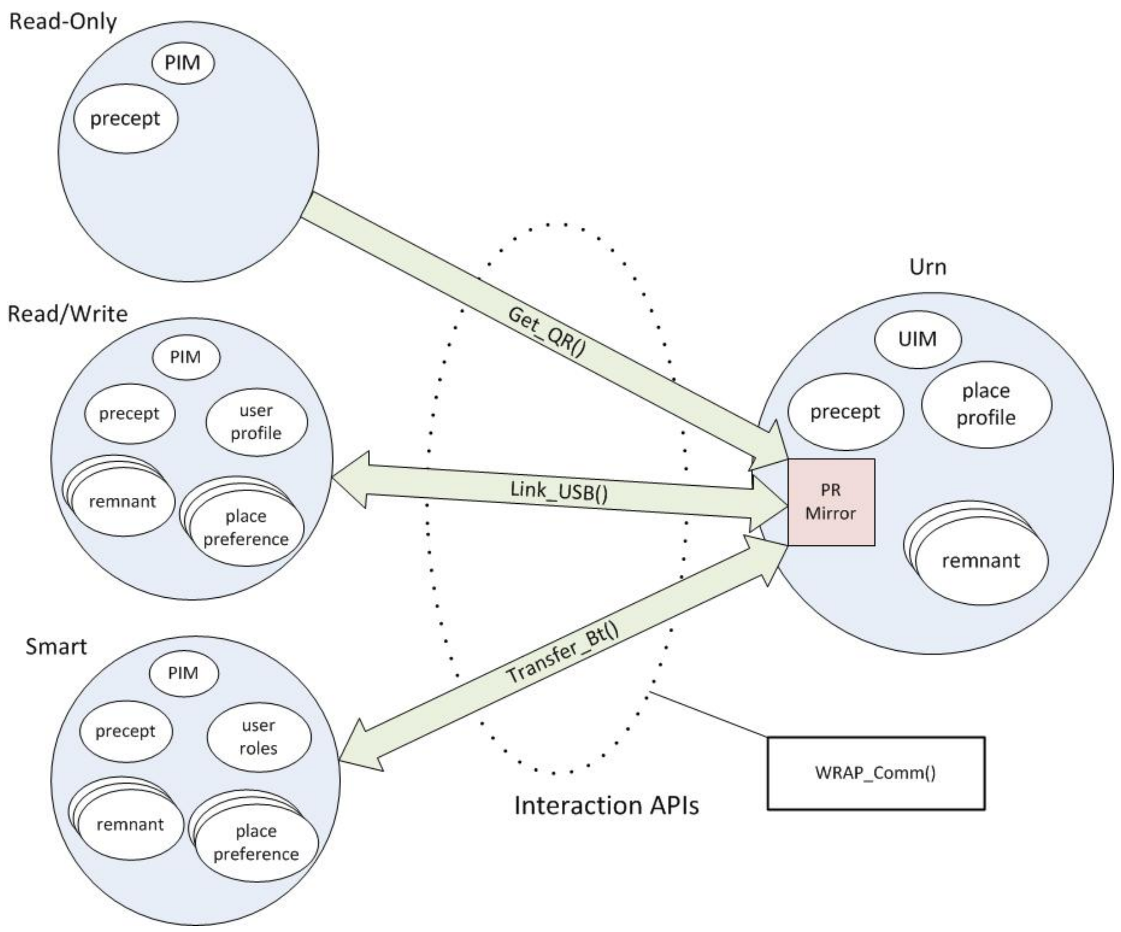


Figure 25 - P&U Transport - Pebble Repository Replication Process

P&U Transport functions and their descriptions are contained in Table 16.

Table 16 – P&U Transport Functions

Urn Functions	**Description**
MIRROR_Load_PR()	Loads a Pebble Repository into the Urn Repository at the beginning of the P&U session
MIRROR_Store_PR()	Sends mirrored image back to Pebble at the end of the P&U session
Smart Pebble Functions	**Description**
MIRROR_Send_PR()	Sends Pebble Repository to the Urn at the beginning of the P&U session
MIRROR_Receive_PR()	Stores mirrored image back onto the Pebble Repository at the end of the P&U session

4.3.2.3 P&U Application Framework

The unique qualities and functions of a particular P&U session are implemented using the P&U Application Framework. Using the APIs in this framework and the Repository's data dictionaries, P&U developers can build Urn services specific to the Urn's place. If developing for a Smart Pebble, user-specific applications controlling roles and privacy can be developed. As depicted in Figure 26, users of Smart Pebbles have the option of selecting a specific user role during their P&U session. This action adjusts the configuration of the Pebble Repository transported to the Urn.

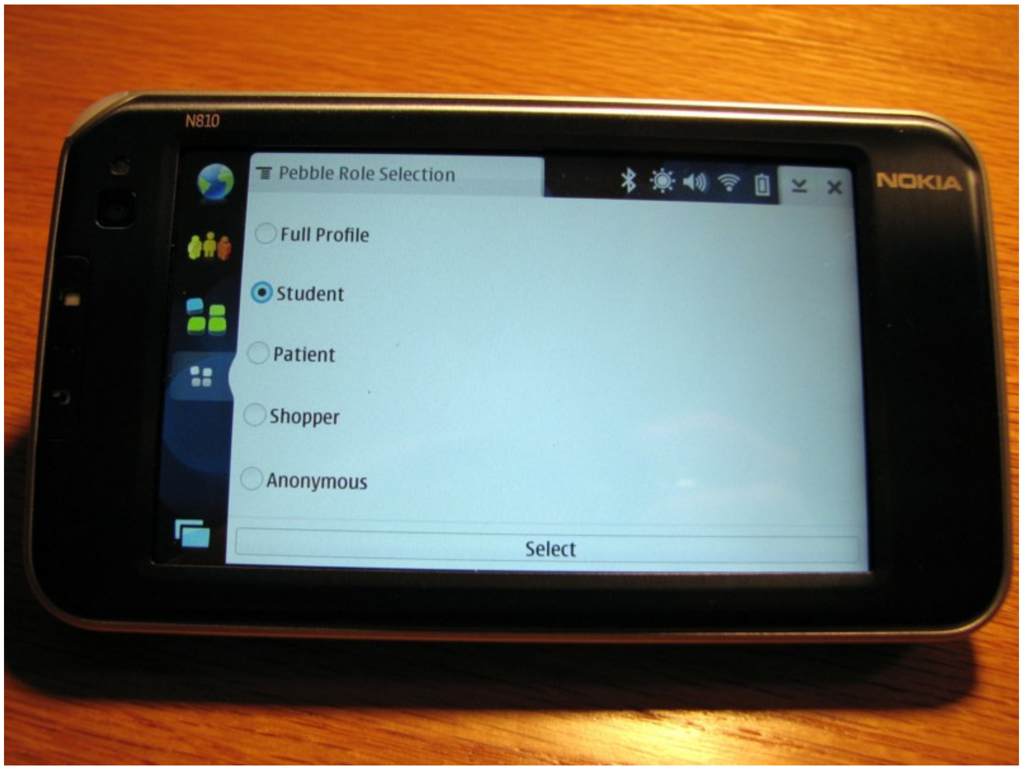


Figure 26 - Role Selection on a Smart Pebble

This action adjusts the configuration of the Pebble Repository transported to the Urn. Figure 27 shows the program flow of a P&U application developed with the Application Framework running on a Smart Pebble.

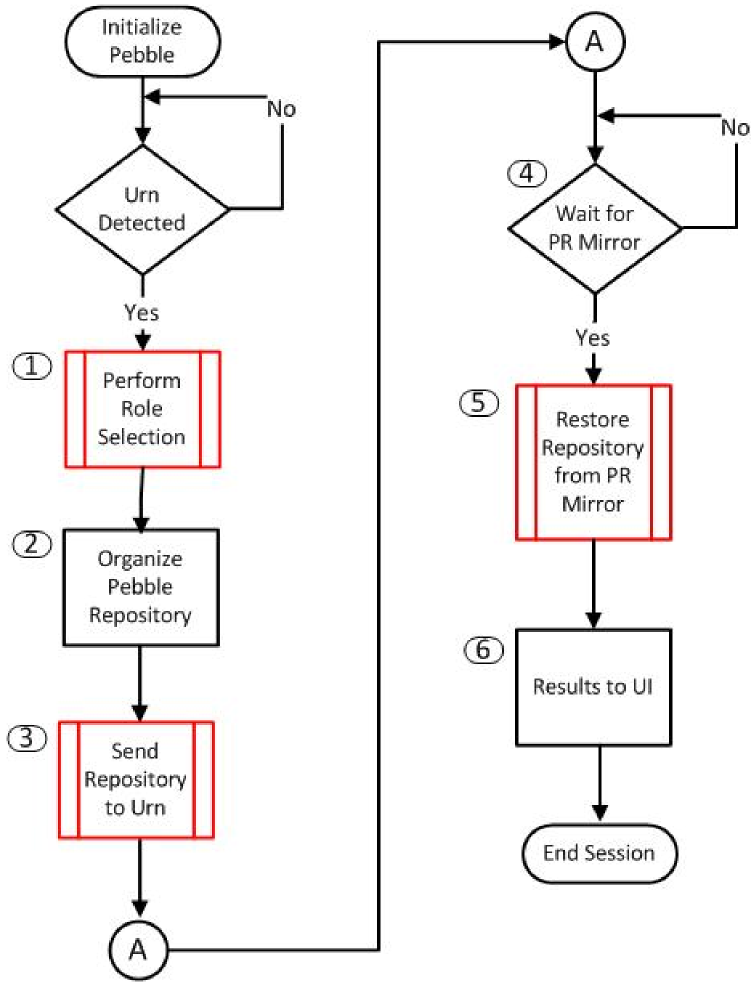


Figure 27 - P&U Smart Pebble Application Flow Diagram

Execution steps of the Smart Pebble application include:

1. The Pebble owner selects the role they wish to assume during the P&U session.

2. The Pebble Controller reorganizes the Pebble Repository to reflect the role selection. Actions include selecting the appropriate user role that will be sent to the Urn during the Pebble Repository replication process. Repository objects not needed for the P&U session are moved to other Smart Pebble storage locations outside of the Pebble Repository directory. Future work includes encryption of Pebble Repository objects not involved in the P&U session.

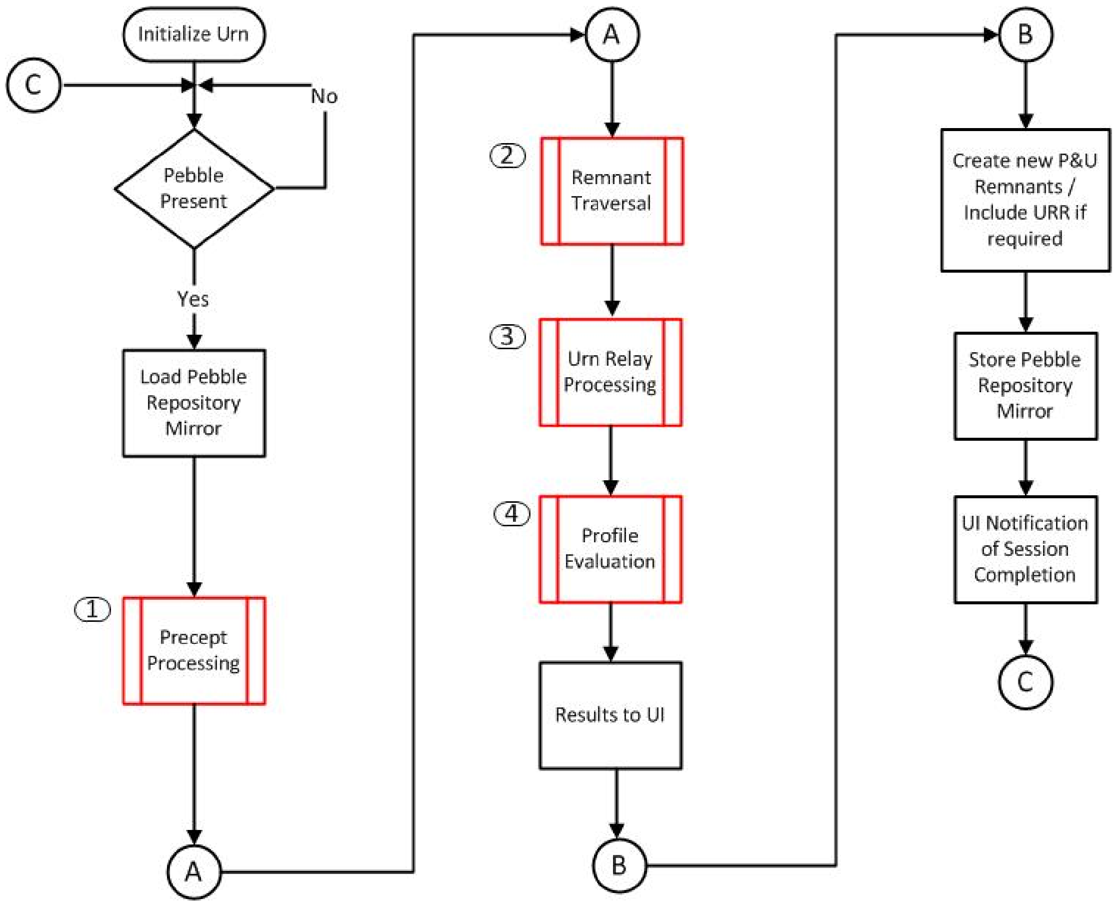
3. The Pebble Repository is sent to the Urn by means of the Pebble Repository replication process.

4. The Smart Pebble waits for updated mirrored Pebble Repository from the Urn.

5. The Pebble restores its Repository with the contents of the mirrored Pebble Repository.

6. Pebble owner notified of end of P&U session.

Figure 28 shows the program flow of a P&U application developed with the Application Framework running on an Urn.



Initialize Urn
C
No
Pebble Present
Yes
Load Pebble Repository Mirror
1
Precept Processing
A
A
2
Remnant Traversal
3
Urn Relay Processing
4
Profile Evaluation
Results to UI
B
B
Create new P&U Remnants / Include URR if required
Store Pebble Repository Mirror
UI Notification of Session Completion
C

Figure 28 - P&U Urn Application Flow Diagram

An assessment of opportunistic situations is performed during the following phases of execution:

1. Precept Processing – This step performs a simple service discovery and delivery operation. The user's service requests contained in the Pebble's precept data element are compared against the available services contained in the Urn's Precept data element. If a match is detected, the service is executed on the Urn by the Urn Controller and results presented to the user on the Urn UI.

2. Remnant Traversal – The remnant data elements on the Pebble and Urn can be analyzed to determine prior P&U interactions. Types of interactions include identifying the number of times that a particular Pebble has interacted with a particular Urn or generating a list of Pebbles encountered by an Urn on a given date. Remnant traversal allows a P&U system to maintain a sense of persistence. With this, an Urn could potentially offer a reconstruction service leveraging previous experiences of people, places and events.

3. Urn Message Relay – Messages including state information can be carried from one Urn to another by a Pebble. This feature supports synchronization of stand-alone Urns. Urn-to-Urn messages reside, or "piggy-back", in the Urn Remnant placed in the Pebble Repository during P&U sessions. Another function of a message would be to trigger a user event in the current P&U session based on a message from an Urn encountered in a prior P&U session. Figure 29 illustrates the Urn Message Relay function. The UMR data structures are defined. Work on the UMR processing function is not complete and the P&U UMR feature is considered future work.

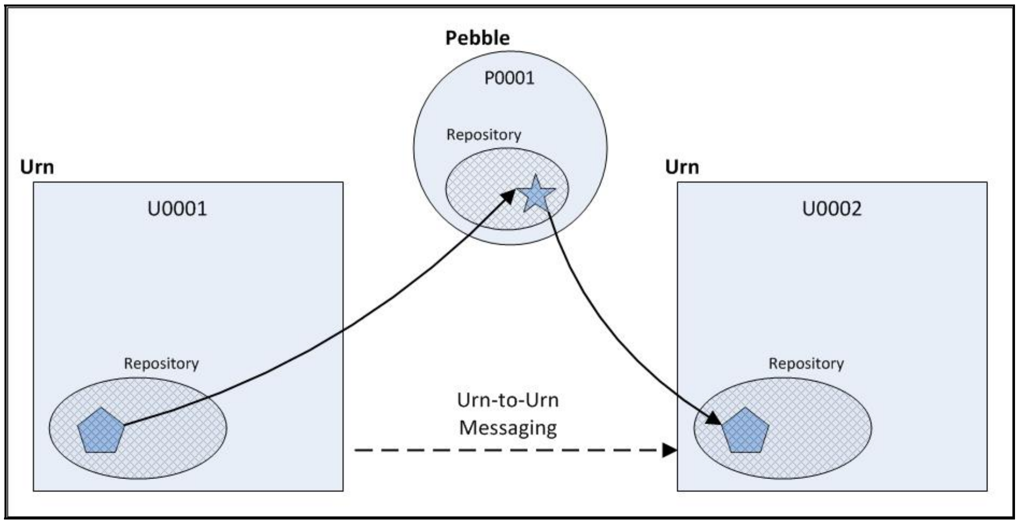


Figure 29 - Urn Message Relay

4. Profile Evaluation – The profile and preference information contained in the Pebble and Urn Repositories are compared. Matches are used to tune Urn service responses to the user. An example is a generated list of on-site items that match user characteristics.

Knowledge gained from the use of these functions contributes to the designs of opportunistic engines.

4.3.2.3.1 Components

Within the field of human-computer interaction (HCI), control through feedback is considered one of the most accepted guidelines in the design of interaction [63]. The Pebble owner receives feedback acknowledging correct P&U use from the Urn's user interface (UI). Service request output is also presented by the Urn UI. The Pebble UI on the Smart Pebble allows explicit control by the user in selecting the types of information, such as different user roles, they want presented to an Urn. The Pebble UI is managed by the Pebble Controller. The Urn Controller is responsible for the execution of the P&U states described in Section 4.3.1.

4.3.2.3.2 Application APIs and Internal Functions

The Application Framework APIs are grouped into three categories:

1. Common – These APIs provide data access to the data elements in the P&U Repositories.
2. User Interface – This API controls the Urn UI.
3. External Services – This API provides access to services external to the Urn.

4.3.2.3.2.1 Common Framework

The Urn Common Framework API prefixes are precept (PRE), profile/preference (PRO), remnant (REM), and utility (PUU) on the Urn. Table 17 provides a brief description of the Common Framework APIs.

Table 17 – Common Framework Urn APIs

API	Description
PRE_Precept()	Executes all matching Precepts
PRO_Comp()	Performs comparison of user and place profiles
PRO_Update()	Updates user profile in Pebble Repository
REM_Get_Remnant()	Retrieves Urn and Pebble remnants
PUU_UIM()	Retrieves Urn UIM
PUU_PIM()	Retrieves Pebble PIM

Detailed descriptions of these APIs including calling sequences and return values are contained in Appendix B.3. Table 18 describes the Smart Pebble Common Framework API, SP_Select_Role.

Table 18 - Common Framework Smart Pebble APIs

API	Description
SP_Select_Role()	Performs role selection

Detailed descriptions of these APIs including calling sequences and return values are contained in Appendix B.3.1.

4.3.2.3.2.2 User Interface

Table 19 describes the User Interface API responsible for input/output control of the Urn UI.

Table 19 - P&U User Interface APIs

API	Description
UI_Text_Out()	Displays text on Smart Pebble or Urn display

Detailed descriptions of these APIs including calling sequences and return values are contained in Appendix B.3.1.

4.3.2.3.2.3 External Services

External services such as e-mail and remote data access are provided by the External Services API described in Table 20.

Table 20 - P&U External Services APIs

API	Description
ES_Send_Email()	Urn external service – send email

Detailed descriptions of this API including the calling sequence and return values are contained in Appendix B.3.1.

4.3.2.3.2.4 P&U Developer Tools

Tools designed to assist the P&U developer with Pebble and Urn creation are described in Table 21. Source code of the P&U framework and developer tools is contained in a software repository at https://sourceforge.net/projects/pebblesandurns/.

Table 21 – P&U Developer Tools

Function	Description
Create_UIM()	Create Urn initialization message
Print_UIM()	Print Urn initialization message
Create_PIM()	Create Pebble initialization message
Print_PIM()	Print Pebble initialization message
Create_Urn_Precept()	Create Urn precept
Print_Urn_Precept()	Print Urn precept
Create_Pebble_Precept()	Create Pebble precept
Print_Pebble_Precept	Print Pebble precept
Create_Place_Profile()	Create place profile
Print_Place_Profile()	Print place profile
Create_User_Profile()	Create user profile
Print_User_Profile()	Print user profile
Print_Pebble_Remnant()	Print Pebble remnant
Print_Urn_Remnant()	Print Urn remnant

Appendix D describes the use these developer tools to construct a Pebble and an Urn.

4.4 Summary

This chapter described the rationale used in expanding the P&U conceptual view presented in Chapter 2 to a create an early instance of a P&U architecture. Research activities described in Section 4.2.3 have produced a more complete P&U development framework, as was described in Section 4.3.

The evolved P&U framework of Section 4.3 addresses, at least in part, Research Questions 1.1 and 1.2. For Research Question 1.1, the isolated communication layer of the initial architecture was extended to support different Pebble classes further supporting near, situated, tangible interactions. For Research Question 1.2, the expanded Repository elements and associated data operations are used to investigate ways opportunistic situations can be identified during a P&U session.

Chapter 5 builds on an initial P&U design by extending work to include several P&U designs based on the P&U framework described in Section 4.3. Chapter 5 discusses the implementation and testing of the P&U instances based on these designs.

Chapter 5: Instantiation of the Pebbles and Urns Framework

This chapter describes the implementation of Pebbles and Urns prototypes based on the P&U designs presented in Chapter 4. The approach in implementing a prototype based on the initial P&U design described in Section 4.2 is discussed in Section 5.1. Section 5.2 introduces the construction of prototypes based on the evolved P&U design described in Section 4.3. Prototypes based on each of the Pebble classes described in Section 2.2.1 are presented in Sections 5.3 through 5.5. Section 5.6 examines overall testing and performance results for the prototypes. Section 5.7 provides a summary of findings and conclusions resulting from the instances of P&U implementations described in this chapter.

5.1 Initial P&U Prototype – Pebble of Interest

As discussed in Section 2.3, the interaction between a Pebble and an Urn is a focal point of the initial phase of this research. This first P&U prototype is an implementation of the portion of the P&U architecture that is highlighted in Figure 30. The Pebble and the Urn are endpoints of this specific implementation.

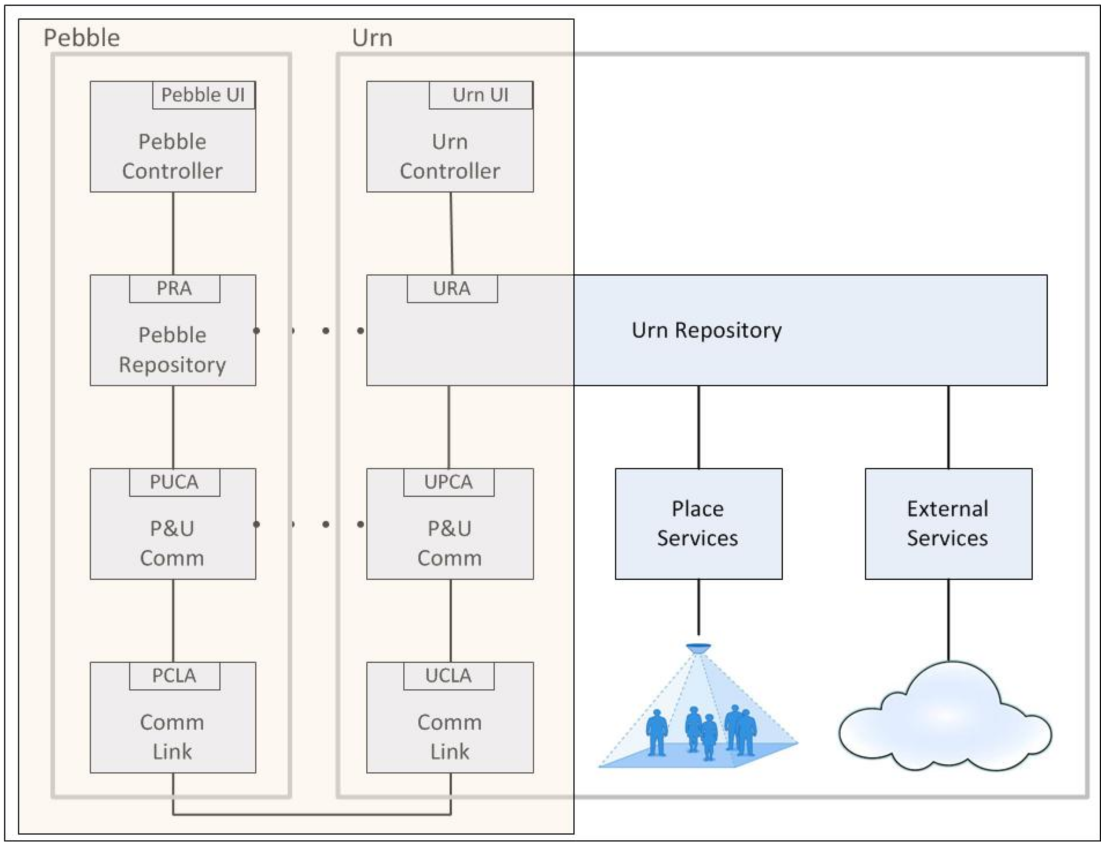
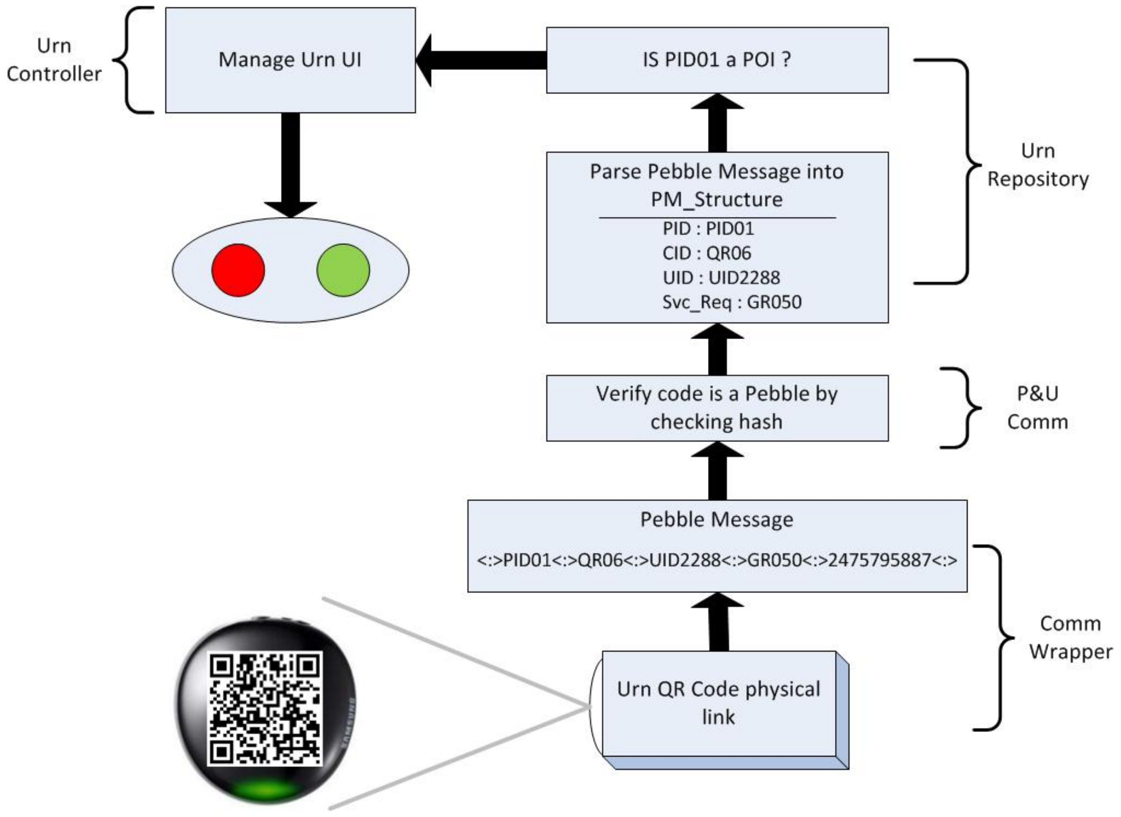


Figure 30 - Components of the Initial P&U Prototype

Two design attributes of the P&U architecture described in Section 4.1.1 are: (i) the enforced near, situated interaction of the Pebble and Urn; and (ii) the isolation of the Urn from its physical communication mechanism. These two attributes are demonstrated in this P&U prototype implementation.

The prototype is based on a Read-Only class of Pebble interacting with the Urn via a short-distance, scanning interface. Not all components of the Pebble stack are implemented in this prototype given the simple characteristics of this Pebble. The function of this relatively simple P&U prototype is to indicate the presence of a "pebble-of-interest" (POI). A POI is a Pebble possessing attributes that make it a suitable participant in a specific Pebble and Urn interaction. The Urn can verify a POI by comparing the Pebble's PID against a list of known POIs stored in the POI table in the Urn Repository. Using the blocking approach described in

Section 4.2, the Urn waits for the presence of a Pebble. The Urn UI flashes green when a POI is detected as being present. The control flow of the P&U prototype is shown in Figure 31.



Urn Controller
Manage Urn UI
IS PID01 a POI ?
Urn Repository
Parse Pebble Message into PM_Structure
PID : PID01
CID : QR06
UID : UID2288
Svc_Req : GR050
Verify code is a Pebble by checking hash
P&U Comm
Pebble Message
<:>PID01<:>QR06<:>UID2288<:>GR050<:>2475795887<:>
Comm Wrapper
Urn QR Code physical link

Figure 31 - Control Flow of the Pebble of Interest P&U Prototype

The next section describes the coding details of the P&U prototype.

5.1.1 Coding Details of the Initial Prototype

The code for the P&U prototype (just the Urn, in this case) is written in Python 2.5.2 [64] and runs in a Lubuntu 11.10 Unix environment [65]. The Urn's UI display is simulated using the Python GUI toolkit wxPython [66]. The wrapper communication link uses the barcode reader library, ZBar [67]. The Urn in the prototype is represented by a small laptop computer with built-in camera. The QR Pebble, shown in Figure 32, is a one-inch cube with QR code surfaces. The QR code surfaces were generated from the goQR.me website [68]. The QR code format used in the prototype is a 250 × 250 pixel text code with 2-pixel margins. Data encoding is UTF-

8 and the error correction code is "L" [69]. Data pixel and background colors are of the 0xhhhhhhhh format. The generated QR code image is transferred to a 1-inch × 1-inch Microsoft Word template. A 1-point solid line border is applied to the image after reducing image size to 0.94 inch × 0.94 inch.

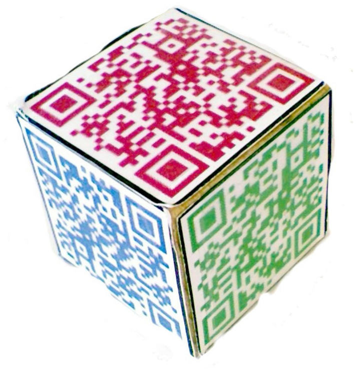


Figure 32 - QR Pebble

Figure 33 is the state diagram for the prototype Urn. The diagram illustrates the blocking read technique used in the Pebble and Urn interaction. The Urn waits for the presence of a QR code. Once detected, the QR code is analyzed to determine if it is a valid Pebble. If a valid Pebble is present, updates to Urn Repository tables occur. If the Pebble is a POI, the Urn's UI presents a green display. If the Pebble is not a PI, then the Urn's URI presents a red display. After this event, the Urn Controller initiates another read and waits for another Pebble.

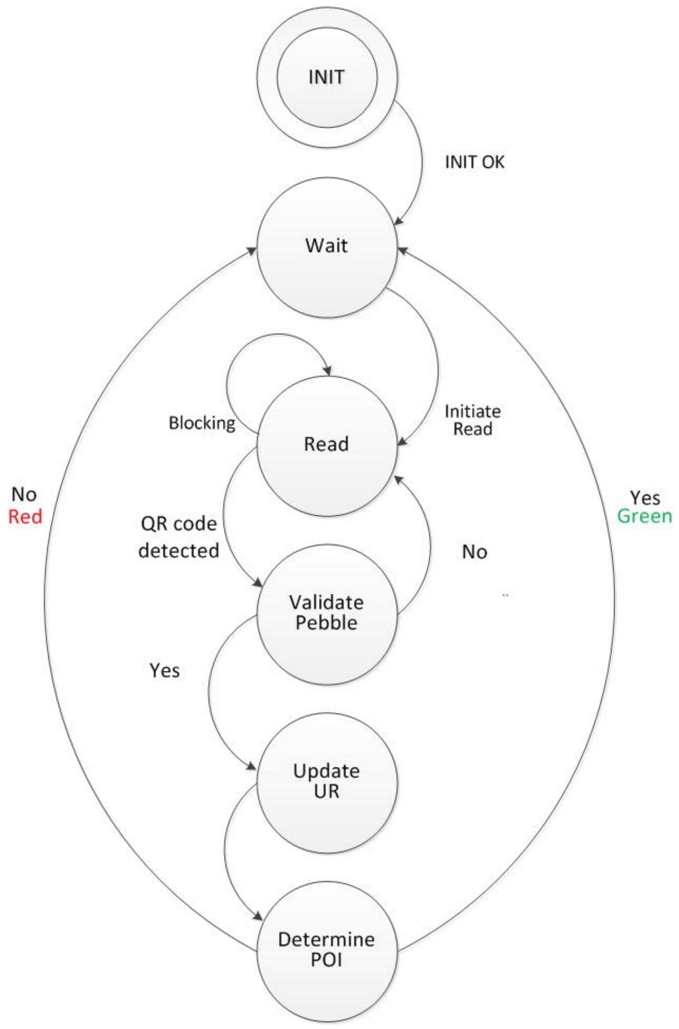


Figure 33 - P&U Prototype State Diagram

The code module for each component of the prototype uses the functions and APIs described in Section 4.3. The subset of the P&U architecture used in the prototype is shown in more detail in Figure 34. Hashed areas of the Pebble architecture represent components and connectors not present in the QR Pebble. The Communications Link in this prototype is the short-range visual scanning of a QR image on a Pebble by the Urn's camera.

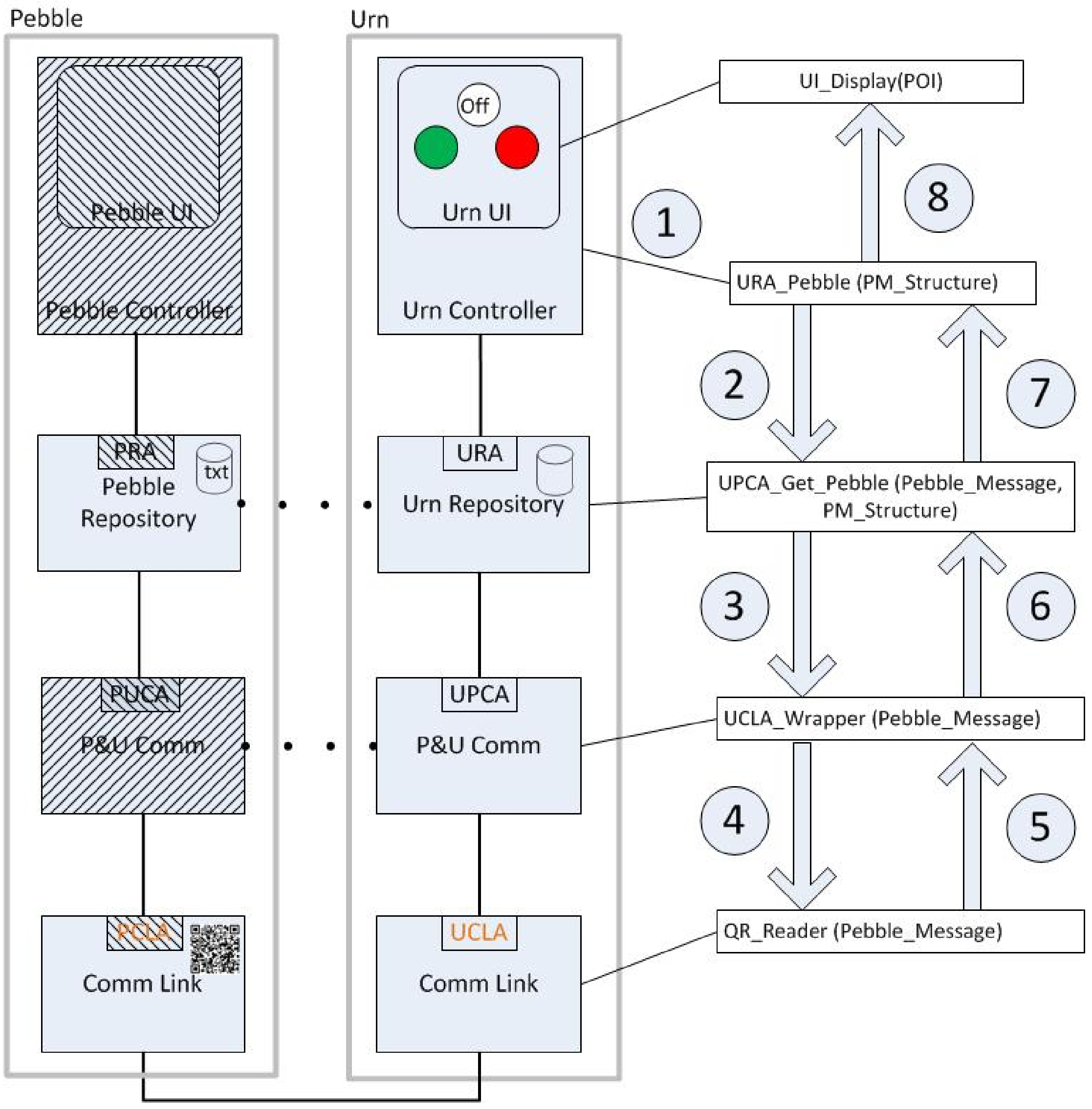


Figure 34 - P&U Prototype Architecture

The call sequence during execution of the P&U prototype is as follows.

Step 1. The Urn Controller initiates the “wait for Pebble” by calling the Urn Repository API function URA_Pebble(PM_Structure). At this point, PM_Structure is empty.

Step 2. The Urn Repository pushes the Pebble request down the stack with a call to UPCA_Get_Pebble(Pebble_Message,PM_Structure). Pebble_Message is a null string.

Step 3. The P&U Communication component calls the wrapper code, which invokes the PMI method needed to interact with the designated Pebble class.

Step 4. In the POI prototype, the wrapper code invokes the QR reader function.

Step 5. The QR reader function blocks until a QR code is read. The QR code text is returned in the Pebble Message string.

Step 6. The UCLA wrapper code generates a hash of the Pebble message body and compares it to the hash message trailer. If the hash value is valid, the Pebble Message is parsed into the PM_Structure. PM_Structure is returned to the Urn Repository.

Step 7. The Urn Repository checks the PID portion of PM_Structure for POI status. Status is returned to the Urn Controller.

Step 8. The Urn Controller displays the POI status on the Urn UI. The call sequence repeats after the POI status is displayed.

5.1.2 Outcomes and Discoveries

This section describes the outcomes and discoveries realized during the implementation of the Pebble of Interest prototype.

The P&U prototype executed as described in Section 5.1. Key elements of the initial P&U architecture were successfully incorporated into the prototype's design: a layered composition with APIs, a PMI-style tangible interface, and the Urn's isolated physical communication layer. The prototype validated the design.

The following list summarizes key points learned during the implementation of this P&U prototype. The limitations observed are, in part, motivation for the evolved P&U design.

- The Pebble message communication format as implemented in the prototype is incomplete. The use of simple token delimiters ('<:>') of message fields does not support the stable communication requirement stated in Research Question 2. P&U message design requires more robust formats, possibly derived from TCP or a similar reliable protocol.
- Variability in QR code characteristics used on the QR Pebble affected the Urn's ability to read the Pebble. For example, the Urn could not read low-contrast (yellow/white) color combinations. While this may be considered a QR code issue, the use of QR codes with a QR Pebble also makes it a Pebble design issue.
- The prototype Urn was unable to read multiple QR Pebbles simultaneously. The blocking QR reader code returns when one QR code is recognized. The P&U interoperability task described in Section 6.2.1.1 will address design considerations to support concurrency.
- Data structures used in the prototype are simple. More complex structures will be required as concepts such as state management are added to the P&U architecture.
- The wrapper code performing the QR code read represents a solution to a specific PMI interface need. Interface standardization is required to truly isolate the Urn's physical communication layer from its upper layers.
- As a general point, challenges exist when deciding the "division of labor" distributed across the components of a layered architecture. Responsibilities of the P&U components will change as future designs are considered.

These initial findings contributed to the evolved design of the P&U framework. This iterative refinement process has continued with the design, implementation and analysis of additional P&U prototypes. Details of these prototypes are covered in the following sections.

5.2 Evolved P&U Prototypes

This section describes the approach used in constructing prototypes based on the evolved P&U framework. Each of the following prototypes interacts with a Pebble instance from one of the Pebble classes discussed in Section 2.2.1. Section 5.3 presents a P&U system prototype that uses a Read-Only Pebble. The prototype in Section 5.4 uses the Read/Write Pebble. Section 5.5 describes a P&U system based on a Smart Pebble. Figure 35 illustrates the Pebble instances used in the P&U prototypes. The Read-Only Pebble uses a QR code interface. The Read/Write Pebble is based on a USB "thumb drive". The Smart Pebble is implemented on a Nokia N810 handheld Internet appliance and uses Bluetooth as its P&U interaction technology.

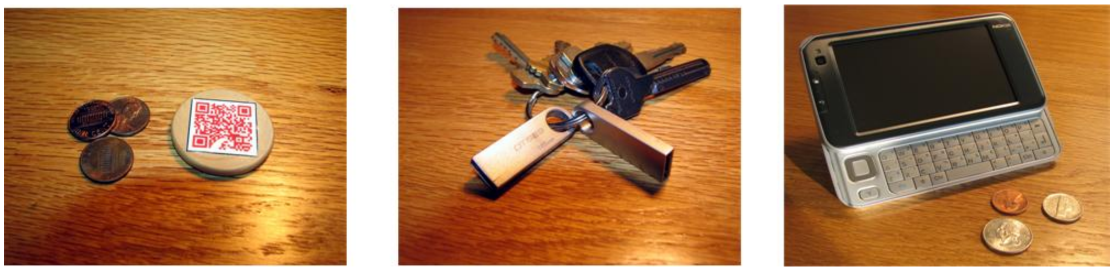


Figure 35 - Pebble Instances (R-O:QR, R/W:USB, Smart:BT)

While an actual Urn in a P&U system may be designed and implemented to interact with all Pebble classes, these prototypes each interact with Pebbles of a single class. The prototypes address aspects of the research questions presented in Section 1.2 by demonstrating the advantages and disadvantages of each of the different Pebble classes. Each prototype is built upon the P&U framework described in Section 4.3 using the API's and developer tools presented in Sections 4.3.2.1, 4.3.2.2.2, 4.3.2.3.2.1, and 4.3.2.3.2.4. Construction of the prototypes also identifies fundamental application development decisions the P&U developer must make each time a P&U system is built. With the goal of building a P&U system that creates opportunistic situations, the developer considers the design of the Pebble within the feature set of its Pebble class, the design of the applications within the Urn's Application Framework and the interactions between these two elements.

The Urns depicted in the prototypes described in Sections 5.3.3, 5.3.4, and 5.3.5 are built on the P&U framework. Each Urn's Comm Link and Transport layers support the different Pebble class interaction technologies by means of the Interaction Framework functions described in Section 4.3.2.1. Urn Repository data elements, as described in Section 4.3.2.2.1, are the same regardless of interactions with Read-Only, Read/Write, or Smart Pebbles. Different content and contextual data in the Urn Repositories include Precepts comprising Urn services, Remnants containing interaction snapshots specific to a place, and place profiles. Specific Urns in the prototype scenarios are the Classroom Urn located in a classroom, the Gym Urn in a public workout facility and the Grocery Urn in a local grocery store.

Common Precept Names (CPN) denoting requested services reside in the Pebble Repository's Precepts data structure. Corresponding CPN's stored in the Urn's Repository Precepts area identify services provided by the Urn. Urn CPN's link to the corresponding programs and data, located in the Urn's Precepts directory, that define and provide the available services. Urn Repository Precepts used in the scenarios are described in Table 22.

Table 22- Urn Repository Precepts

Classroom Urn Precepts		
CPN	*CPN Executable*	*CPN Data*
CR01	group_assign.py	---
CR02	assignment_collect.py	---
CR03	new_assignment.py	---
Gym Urn Precepts		
CPN	*CPN Executable*	*CPN Data*
G01	gym_access.py	---
G02	avail_equip.py	gym_sensors
G03	current_resist.py	user_pref
Grocery Urn Precepts		
CPN	*CPN Executable*	*CPN Data*
GS01	item_loc.py	store_inventory
GS02	item_price.py	store_inventory
GS03	award_coupon.py	---

The prototypes are written in Python 2.6.2. Urns in the prototypes are implemented on a Dell Mini 9 netbook (1.60 GHz processor, 2G RAM, 3991 BogoMips) running the Linux Ubuntu 12.10 operating system. The Smart Pebble is implemented on a Nokia N810 internet tablet

(400 MHz TI OMAP 2420 ARM processor, 256 MB + 2 GB Flash, 164 BogoMips) running the Maemo 4.1 operating system (a Unix/Linux variant).

A scenario describes how the use of these prototypes affects a user's experience while performing several typical daily activities. Each of the following prototype sections contain a scenario about the use of a particular Pebble class, application considerations, coding details, quantitative analysis of prototype components and a discussion of outcomes and discoveries. Prototype scenarios are based on the following scenario describing an environment in which P&U does not exist.

> *It's Wednesday evening, and Alice has just finished dinner. Now it's time to run an evening of errands. Alice makes sure she has her grocery list and her pass card to the gym. Tonight is class night, so she also takes her textbook off the nightstand as she leaves her apartment. Her first stop tonight is class.*
>
> *Alice enters the classroom and takes a seat. Starting class, the instructor asks the students to turn in their homework. As she looks through her textbook, Alice realizes she left her homework on her study desk. Class continues as the instructor assigns each student to a group. Alice leaves her seat to go join her group members in another area of the classroom. Class ends, and Alice, still disappointed about the forgotten homework assignment, heads to the gym.*
>
> *Arriving at the multi-story exercise facility, Alice enters using her pass card. Her exercise routine consists of a sequence of exercises using different equipment located throughout the facility. On busy nights, Alice has to spend time searching for equipment not in use. Completing her exercise routine, Alice leaves for her last stop of the evening, the grocery store.*
>
> *Alice enters the large FoodShop store with grocery list in hand. Taking a cart, Alice begins to shop for the items on her list. She quickly realizes, much to her dismay, that FoodShop has once again rearranged store items to different*

locations in the store. Alice continues to now search for the items on her grocery list. She becomes more frustrated when she discovers, after finding the correct shelf, that the store is out of an item. Finally locating most of the items, Alice proceeds to the grocery store checkout. Irritated that her favorite TV show has been on for fifteen minutes, Alice gets in her car and heads home after a somewhat stressful evening of errands.

The following sections describe three alternative scenarios that illustrate how the user's experience can be improved over the scenario above. The alternative scenarios require progressively more capable P&U systems.

5.3 P&U Prototype: Read-Only Pebble

This section describes a P&U prototype based on the Read-Only Pebble class, one of the Pebble classes discussed in Section 2.2.1. Read-Only Pebbles have a limited amount of non-volatile storage and possess no processing capabilities. Pebbles in this class are simple, inexpensive physical objects. Developers would take these characteristics into consideration when developing P&U systems designed to use Read-Only Pebbles. Figure 36 represents the architecture layers and components that support the Read-Only Pebble class.

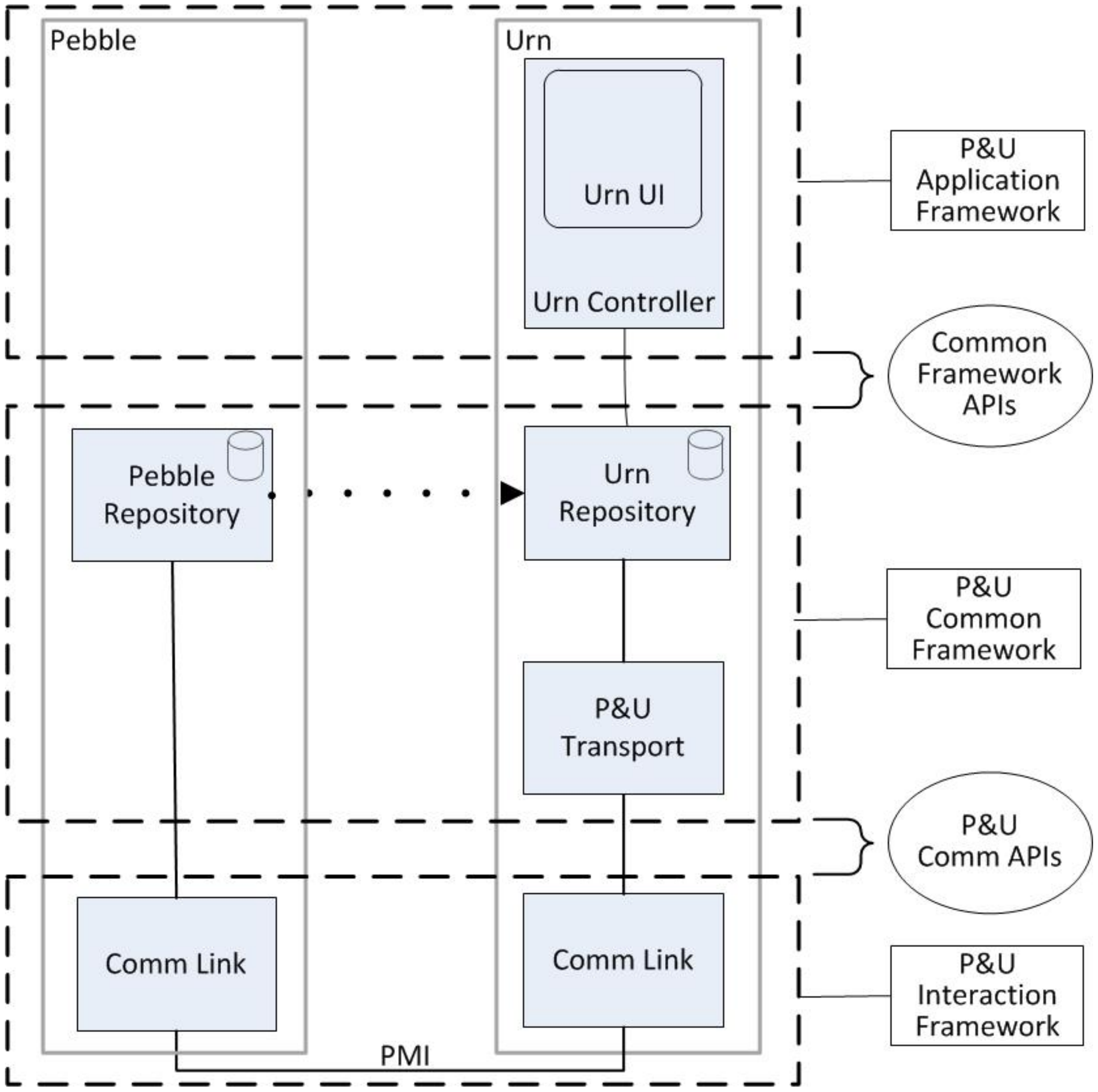


Figure 36 - P&U Framework - Read-Only Pebble

As discussed in Section 4.3.2, the Read-Only Pebble, without a controller or transport component, only provides information to the Urn. Information flow is in one direction from the Pebble to the Urn.

5.3.1 Scenario

The following scenario takes place in an environment with a P&U system that supports Read-Only Pebbles.

It's Wednesday evening, and Alice has just finished dinner. Now it's time to run an evening of errands. Alice makes sure she has her grocery list and her Pebble. Tonight is class night, so she also takes her textbook off the nightstand as she leaves her apartment. Her first stop tonight is class.

As Alice enters the classroom, she holds her Pebble next to the classroom Urn. The characters "3F" appear on the Urn's display. For this particular class session, Alice is in group 3 and assuming a role of facilitator. Alice locates a seat with the other group 3 class members. Starting class, the instructor asks the students to turn in their homework. As she looks through her textbook, Alice realizes she left her homework on her study desk. Class continues as the instructor gives assignments to the different groups. Class ends, and Alice, still disappointed about the forgotten homework assignment, heads to the gym.

Arriving at the multi-story exercise facility, Alice gains access by displaying her Pebble to the gym Urn. Her exercise routine consists of a sequence of exercises using different equipment located throughout the facility. On busy nights, Alice has to spend time searching for equipment not in use. Completing her exercise routine, Alice leaves for her last stop of the evening, the grocery store.

Alice enters the large FoodShop store with grocery list in hand. She shows her Pebble to the grocery store Urn. The Urn acknowledges that it is her fifth visit this month and dispenses a 5% off coupon for this store purchase. Taking a cart, Alice begins to shop for the items on her list. She quickly realizes much to her dismay, that FoodShop has once again rearranged store items to different locations in the store. Alice continues to now search for the items on her grocery list. She becomes more frustrated when she discovers, after finding the correct shelf, that the store is out of an item. Finally locating most of the items, Alice proceeds to the grocery store checkout. Irritated that her favorite TV show *has been on for ten minutes, Alice gets in her car and heads home after a somewhat stressful evening of errands.*

5.3.2 Application Considerations and Coding Details

The scenario presented in Section 5.3.1 contains the following P&U service interactions.

- Assignment of a group designation to a student in a classroom setting
- Access request to a physical space
- Opportunistic interaction enforcing user participation

Each of these interactions, involving the same Pebble owner and her Pebble, consist of different service applications running on different Urns located in different places. The P&U developer uses the P&U framework to build the different Urns and service applications. Construction of Pebble instances based on the different Pebble classes is also guided by the P&U framework. For example, the structure and content of Pebble Repositories are modeled after comparable Urn Repository components.

The classroom application assigns a group and role designation to the student as they enter the classroom and present their Pebble to the Classroom Urn. Assignment is configurable based on either a random selection without replacement from a predefined pool of group and role designations, or programmatic selection based on student identification number and instructor preference.

The gym application unlocks the gym door when the user presents her Pebble to the Gym Urn. Access is granted if the user's userid on their Pebble matches an entry in the gym's active membership. A prior registration process adds the user's userid to the gym's active member list.

The grocery store application running on the Grocery Urn demonstrates the use of Remnants to present an opportunistic situation to the user. A discount coupon is presented to the user if they demonstrate a certain activity pattern relative to their interaction with the Urn's place. An example is the award of a store discount coupon if the user has visited the store a certain number of times over a certain period. Randomization strategies can be incorporated in the coupon award application to protect against user behaviors intended to take advantage of the system.

The QR code scanning interface used in this prototype is similar to that used in the initial P&U prototype described in Section 5.1. Construction of the Read-Only Pebble used in this prototype follows the QR code generation guidelines presented in Section 5.1.1. The control flow diagram of this P&U prototype is shown in Figure 37.

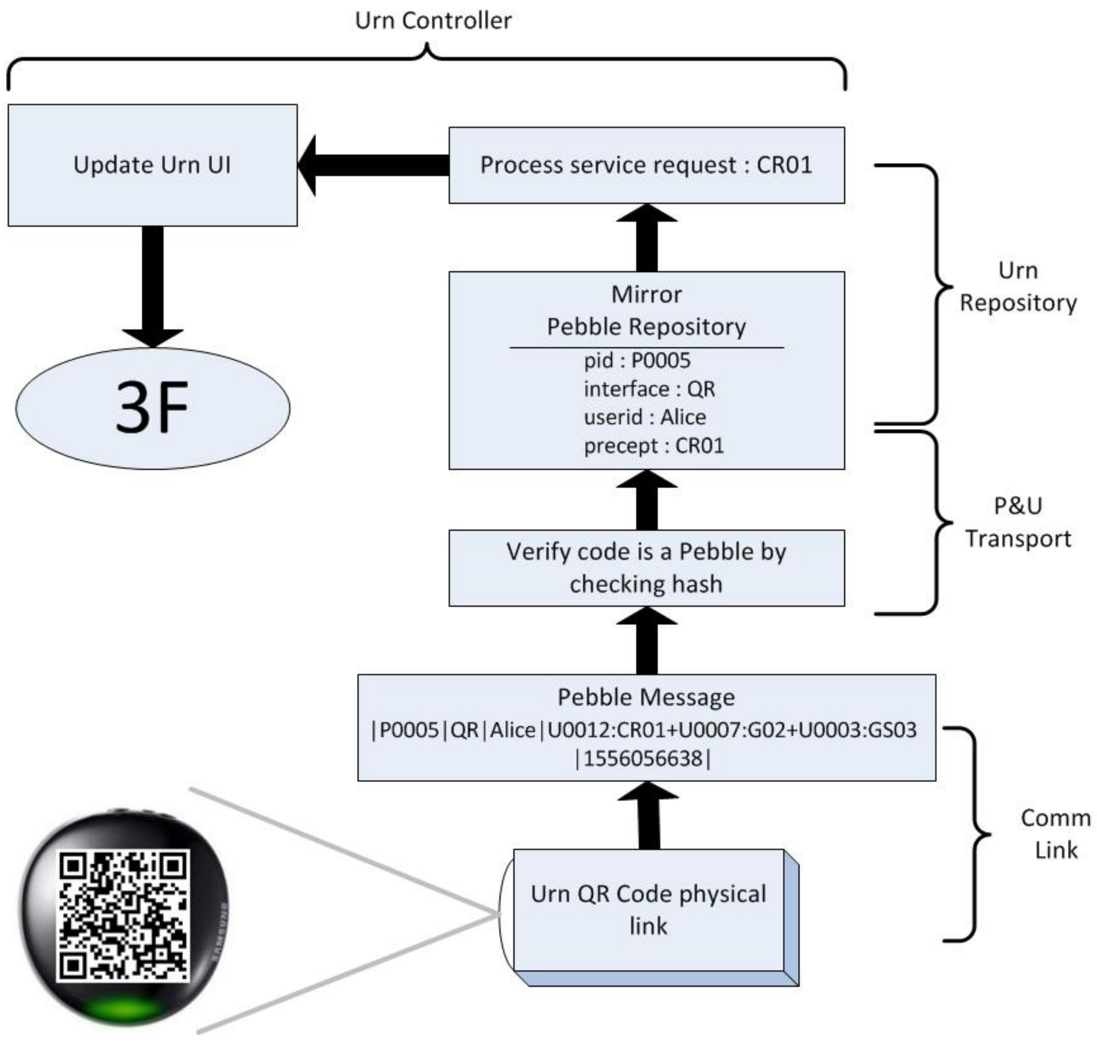
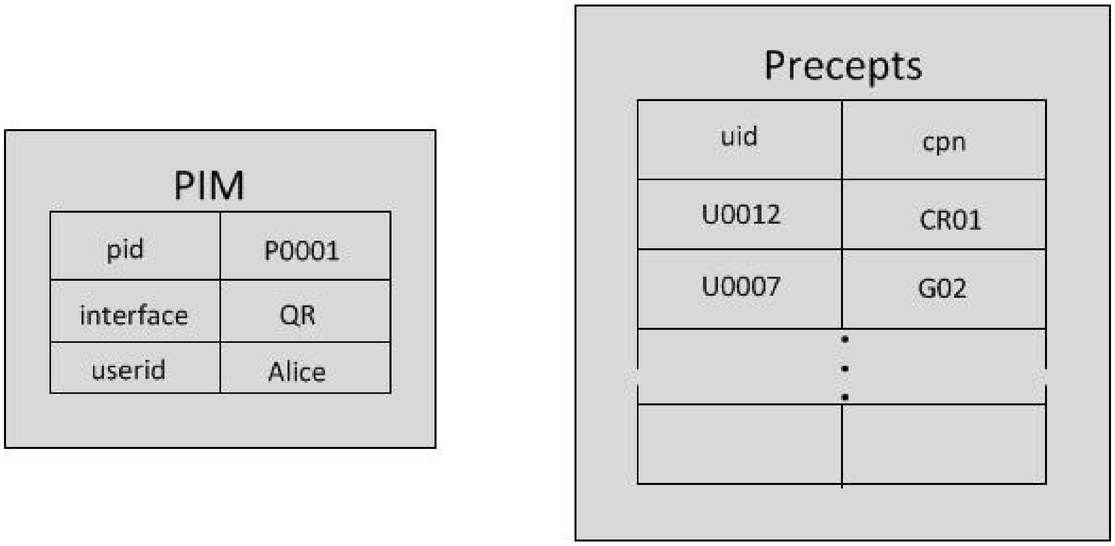


Figure 37 - Control Flow - Read/Only Pebble

Due to storage constraints, the Read-Only Pebble Repository contains a subset of the Repository elements described in Section 4.3.2.2.1. The Read-Only Pebble Repository does not contain Urn Remnants or user preference data elements. Read-Only Pebble Repository's Precepts data element contains only CPNs, lacking space for the additional CPN data element. In addition, the Read-Only Pebble Repository data elements cannot be updated by the Urn during a P&U session. As a result, P&U systems using Read-Only Pebbles do not support inter-Urn messaging or user preference/place profile matching. Lack of these functions, described in Section 4.3.2.3, could reduce the Urn's effectiveness in identifying and delivering opportunistic services. The Read-Only Pebble Repository shown in Figure 38 consists of the PIM and a Precepts list.



PIM

pid	P0001
interface	QR
userid	Alice

Precepts

uid	cpn
U0012	CR01
U0007	G02
• • •	

Figure 38 – Example of Read-Only Pebble Repository

Figure 39 illustrates the QR code text string representing the Read-Only Pebble Repository. The added hash tag is used to assess Pebble validity during data transmission.

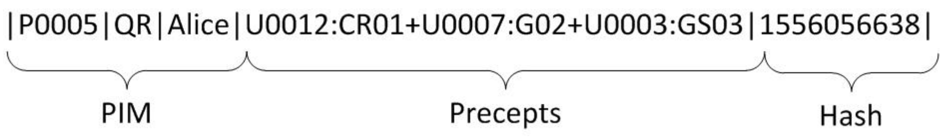
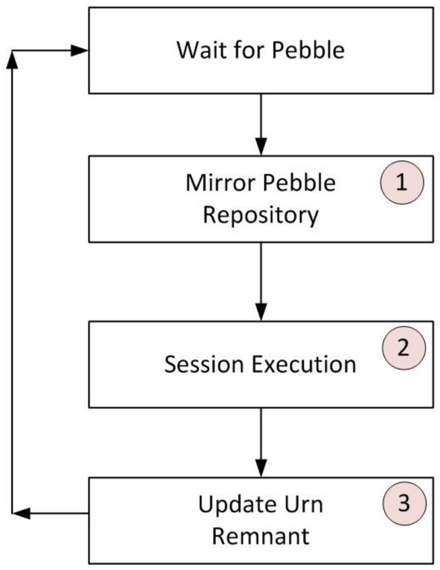


Figure 39- Read-Only (QR) Text Format

Figure 40 shows the program flow and code for applications performing the interactions described in the Read-Only scenario. Step 1 represents the Pebble Repository replication process where the Pebble Repository is transferred to the Urn Repository. P&U session execution occurs in Step 2. In the Read-Only Pebble scenario, Precept processing and Remnant query are actions that occur during session execution. Step 3 creates an Urn Remnant of snapshot data for this P&U session. For Read-Only Pebbles, there is no Pebble Repository update step.



```
# Wait for Pebble
   while True:

# Mirror Pebble Repository onto Urn Repository
   comm.MIRROR.Load_PR()

# Execute Precept Matches
   common.PRE.PRE_Precept('auto’)

# Update Urn Remnant
   common.REM.REM_Put_Remnant(‘urn’)
```

Figure 40 - Read-Only Program Flow and Code

This view of the Read-Only P&U system is expanded in Section 5.3.3 to include descriptions of the basic code blocks of the P&U framework used to build a P&U system.

5.3.3 Testing and Performance Profiling

This section describes testing and performance profiling of the Read-Only Pebble prototype. This study addresses data movement and Urn resource utilization and performance. Profiles and performance metrics of the basic P&U code blocks are obtained by applying a custom profile module. This module uses the efficient Python cProfile module [70] and Python’s logging facility to generate and capture performance metrics of basic P&U code blocks. This profile module can be applied to code segments during runtime and supports module parameter passing and return values. Instrumented P&U code using this profile module can analyze and capture performance metrics from multiple points within a running P&U system.

The P&U code blocks described in Table 23 perform the core functions of the P&U framework. Each code block is configured for analysis by the profile module.

Table 23 - P&U Framework Code Blocks

Generic Code Block	Function
1 WRAP_Comm	• Wrapper code that handles physical communication between Pebble and Urn
2 Mirror_Load_PR	• Replicates Pebble Repository data on Urn • Mirrored Pebble Repository resides in Urn Repository data object
3 PRE_Precept	• Matches Pebble service requests with Urn's provided services • Executes Precept matches
4 REM_Put_Remnant	• P&U session information is captured and stored **a.** Pebble Remnants on Urn **b.** Urn Remnants on Pebble
5 Mirror_Store_PR	• Stores updated mirror Pebble Repository back to Pebble

Table 24 lists the Comm Link and Transport code blocks mapped to their specific Pebble class wrapper functions.

Table 24 - Comm Link and Transport Code Blocks and Wrapper Functions

Code Block	Read-Only	Read/Write	Smart
WRAP_Comm	Get_QR	Link_USB	Transfer_Bt
Mirror_Load_PR	load_QR_mirror	load_USB_mirror	load_Bt_mirror
Mirror_Store_PR	---	store_USB_mirror	store_Bt_mirror

The use of these functions in the Pebble Repository replication process is illustrated in Figure 12 in Section 4.3.2.2.2.

Figure 41 shows the subset of code blocks that make up a Read-Only Pebble P&U instance. This is the program structure of the P&U system depicted in the scenario presented in Section 5.3.1. Code block 1 is named WRAP_Comm and handles the physical communication between Pebbles and Urns. This code adds the PMI functionally described in Section 2.3 to the P&U framework. The module named Get_QR reads the text encoded in a QR code. Code block 2, Mirror_Load_PR, replicates the Pebble Repository onto the Urn. Code block 3 executes any matching Precepts. The last action of this P&U session performed by code block 4(a) is the addition of a Pebble Remnant to the Urn's Remnant data area.

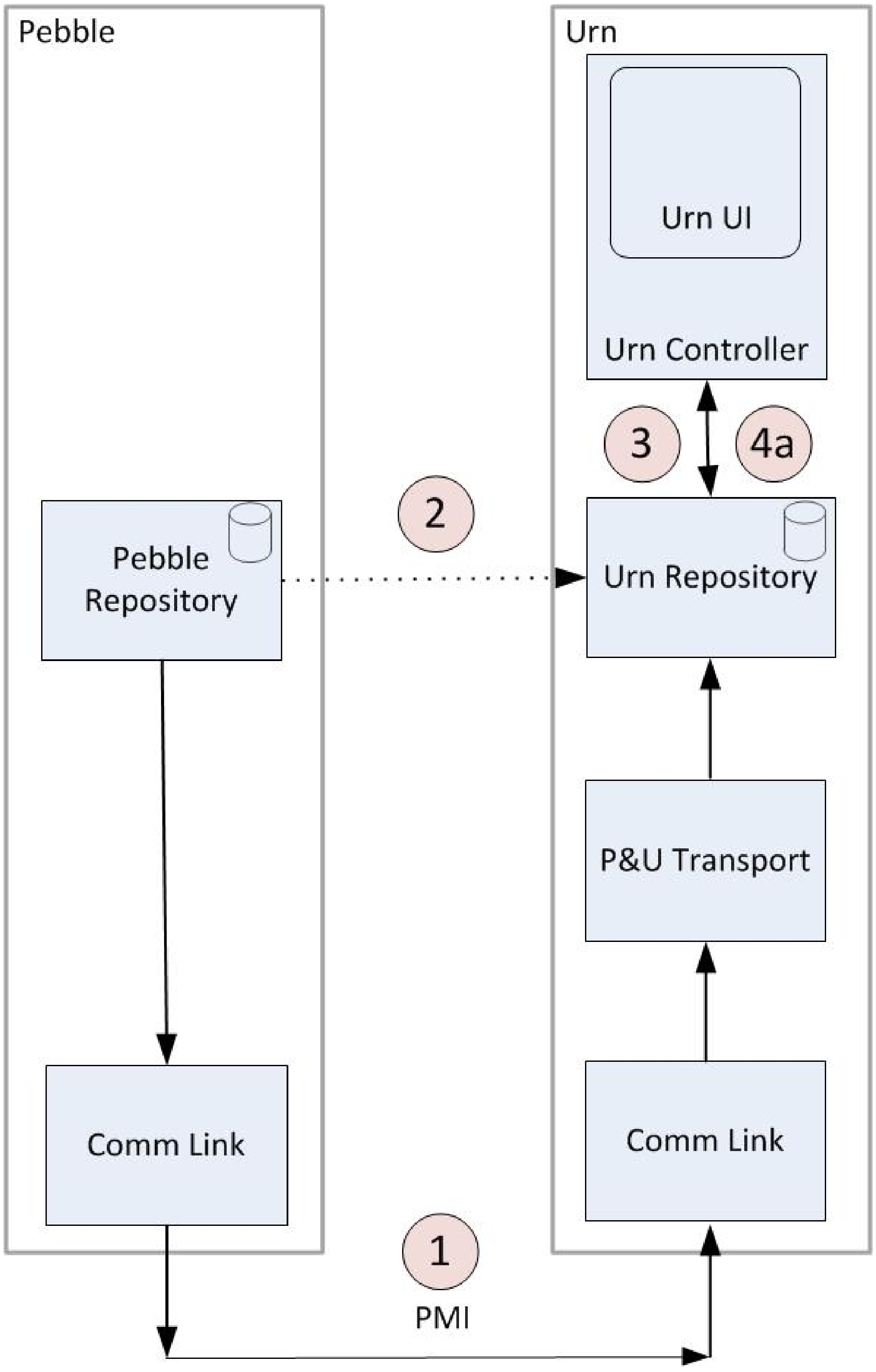


Figure 41 - Code Blocks of the Read-Only P&U Instance

Analysis of the P&U code areas is performed using the profile module described earlier. Performance data is generated by running ten iterations of a P&U session involving the interaction of an Urn and a Read-Only Pebble. The Urn used in this test is the Grocery Urn described in Section 5.3.2. Table 25 summarizes the results from this experiment.

Table 25 - Execution Times of Read-Only Pebble Code Blocks

Generic	Read-Only	Average Execution Time (seconds)	Standard Deviation (seconds)
1 WRAP_Comm	Get_QR	0.878	0.031
2 Mirror_Load_PR	Load_QR_Mirror	0.080	0.008
3 PRE_Precept	PRE_Precept	0.199	0.007
4a REM_Put_Remnant	REM_Put_Remnant	0.024	0.005

The execution times are from code blocks run during the P&U session where Alice interacts with the Grocery Urn using a QR-based Read-Only Pebble. In this experiment, variables associated with the scanning of a QR code are controlled as follows. The Read-Only Pebble is mounted on a fixed support and positioned in front of the Urn's camera. Lighting conditions, Pebble to Urn distance and Pebble movement do not change during the performance measurement iterations.

Operations in the Get_QR step are the decoding of the encoded QR image and a verification check of a hash tag of the decoded text string. The Load_QR_Mirror step parses the decoded QR text string and places the parsed tokens into mirror Repository elements using Python dictionary operators. Step PRE_Precept executes an Urn Precept consisting of an Urn Remnant traversal and match operation using Python dictionary operators. Step REM_Put_Remnant performs a single Python dictionary update operation.

The Get_QR step takes the longest to execute at 0.878 seconds. Removing the hash tag verification step resulted in a Get_QR execution time of 0.856 seconds. The QR code decode operation is the slowest operation in this P&U session. Longer execution times are expected for this step when the Read-Only Pebble is hand-held next to the Urn's camera.

Given that the Rem_Put_Remnant step represents a single Python dictionary update operation of 0.024 seconds, the Load_QR_Mirror step is broken into two string parsing operations (one for PIM elements and one for Precept elements) and two dictionary update operations. Estimated time for the string parsing operation is (0.080 – 2(0.24))/2 or 0.16 seconds.

This particular PRE_Precept step execution time will increase over time as the number of entries in the Urn Remnant structure increases.

5.3.4 Outcomes and Discoveries

This section describes the outcomes and discoveries realized during the implementation, testing, and analysis of the P&U prototype and Read-Only Pebbles.

Execution of the prototype produces results accurately depicting events described in the scenario presented in Section 5.3.1.

The QR-based Pebble is easy to use addressing the "Arrive and Operate" IoT challenge listed in Section 1.1. The need to hold the QR-based Pebble next to the Urn's camera in conjunction with the Urn's fixed location enforces the Pebble owners near, situated, tangible interaction with the prototype Urns. As specified in Research Question 1.1 in Section 1.2, this demonstrates a required characteristic of the P&U architecture.

Services delivered during the P&U sessions match the service domains defined by the different physical locations. For instance, the Grocery Urn provided services relevant to a person buying groceries in a grocery store. Research Question 1.2 in Section 1.2 asks how the P&U architecture supports these service domains.

The Read-Only Pebble did not need to assist with computation and communication tasks requiring a power source during the P&U session. This work was performed by the different Urns. This example of a cyber foraging strategy addresses Research Question 1.3 in Section 1.2.

High-level coding of the Urn prototype using the P&U API can be accomplished in a straight-forward manner. While this is a subjective statement, it is true that the functions could be programmed using the P&U API without needing to include any code that performed lower layer functions of the P&U architecture as described in Section 4.3.2.1. This characteristic of the P&U framework address Research Question 1 in Section 1.2 with regards to supporting IoT and OC implementations.

In consideration of Research Question 2 in Section 1.2, a problematic feature of the P&U prototype is described. The QR-based Read-Only Pebble's limited amount of non-volatile storage reduces the Urn's ability to delivery opportunistic services during the P&U session.

5.4 P&U Prototype: Read/Write Pebble

This section describes a P&U prototype in which Pebbles are based on the Read/Write Pebble class. The prototype Read/Write Pebble based on a USB "thumb drive" has more storage than the prototype Read-Only Pebble based on a QR code, although other physical realizations of a Read-Only Pebble could provide a similar amount of storage as the Read/Write Pebble. The larger storage space present in the prototype allows a larger Pebble Repository that contains additional data elements. In addition, a Read/Write Pebble's storage is rewritable enabling an Urn to update data elements in the Pebble's Repository. These characteristics of the Read/Write Pebble class afford new capabilities within a P&U system as shown in the next section. Figure 42 represents the architectural layers and components that support the Read/Write Pebble class. Information flow is bi-directional between Pebble and Urn.

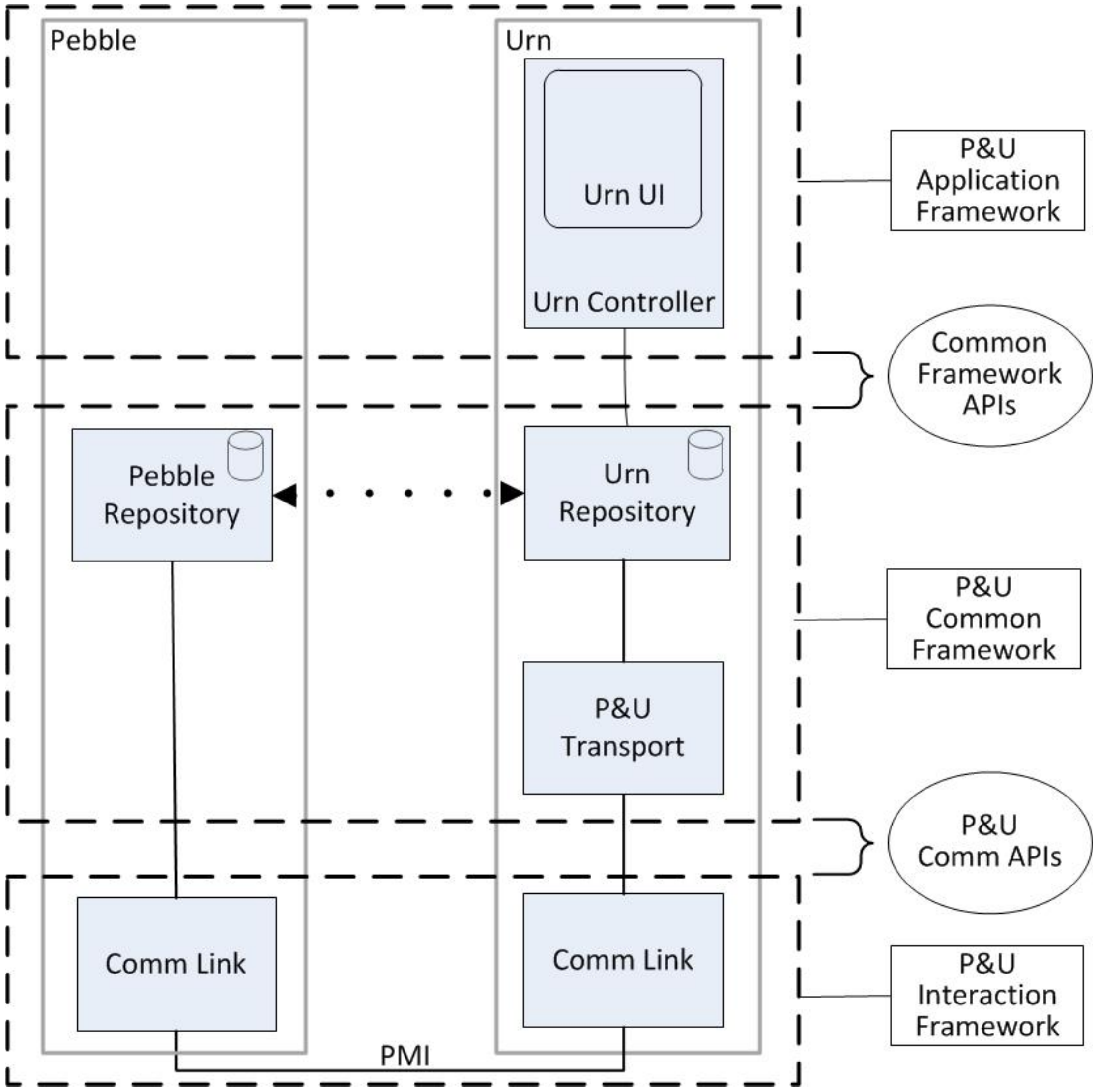


Figure 42 - P&U Framework - Read/Write Pebble

5.4.1 Scenario

The following scenario takes place in an environment with P&U systems that support Read/Write Pebbles.

It's Wednesday evening, and Alice has just finished dinner. Now it's time to run an evening of errands. Alice makes sure she has her Pebble. Tonight is *class night, so she also takes her textbook off the nightstand as she leaves her apartment. Her first stop tonight is class.*

As Alice enters the classroom, she attaches her Pebble to the classroom Urn. The text "Assignment Accepted", "3F", and "Next Assignment Loaded" appear on the Urn's display. Alice acknowledges to herself that for this particular class she is in Group 3 assuming a role of facilitator. See also sees that her assignment due this class period has been moved from her Pebble to the classroom Urn and her next assignment is now on her Pebble. Removing her Pebble from the Urn, Alice continues to the area of the classroom designated for Group 3 members. Class begins as the instructor gives assignments to the different groups. Class ends, and Alice heads to the gym.

Arriving at the multi-story exercise facility, Alice gains access by attaching her Pebble to the door receptacle of the gym Urn. The following message appears on the Urn's display: "1st floor treadmill available, 2nd floor elliptical available", "Current Resistance Level: 5". Her exercise routine consists of a sequence of exercises using different equipment located throughout the facility. The Urn has just displayed available equipment from her exercise profile. The Urn has also reminded Alice that her current resistance setting for the exercise equipment is level 5. Alice heads to the treadmill on 1st floor. Completing her exercise routine, Alice leaves for her last stop of the evening, the grocery store.

Alice enters the large FoodShop store with Pebble in hand. She inserts her Pebble into the grocery store Urn. The Urn acknowledges that it is her fifth visit this month and dispenses a 5% off coupon for this store purchase. The Urn also produces a printout showing each item on Alice's grocery list associated with current store location and price. Aided by this printout, Alice begins her grocery shopping. Alice proceeds to the grocery store checkout already knowing the cost of her shopping trip. Alice gets in her car, turns on the radio, and heads *home after another evening of errands. The reporter on the radio is discussing the growing concerns of identity theft. In spite of their convenience, Alice wonders how safe are these Pebbles? Never mind, she thinks, my favorite TV show begins in 10 minutes.*

5.4.2 Application Considerations and Coding Details

The scenario in Section 5.4.1 describes additional service interactions resulting from the increased capabilities of the Read/Write Pebble. These service interactions include:

- Assistance with operational management of classroom assignments
- Provisioning of user-specific information
- Reduced effort resulting from place-based information

The classroom application now performs P&U interactions involving bi-directional transfer of large data objects. The Read/Write Pebble's Precepts list includes service requests to load completed assignments onto the classroom Urn and to retrieve and store new assignments on the Pebble.

The gym application accesses user preference and place profile information to assist the Pebble owner in locating available exercise equipment. The Pebble owner's current equipment resistance setting is retrieved from a user/place profile and displayed.

The Precept structure in the Read/Write Pebble Repository associates data with Common Precept Names. In the grocery store application, this data is the grocery list. The Grocery Urn has access to grocery item inventory data stored in its Precepts directory. The GS01 and GS02 Precepts merge grocery list and item inventory producing grocery list item locations and prices.

The Read/Write Pebble is implemented on a conventional USB "thumb drive". The Pebble Repository on the USB drive is based on the same data structures and data operators as those used with Urn Repositories. P&U developer tools are used to build both Pebble and Urn Repository instances. This Pebble is associated with an Urn by physically plugging it into a

USB connector on the Urn. This physical connection supports Research Question 1.1 by demonstrating the use of a tangible interface and the enforced physical proximity between Pebble and Urn. When the Read/Write Pebble is plugged into the Urn, the Linux operating system's device manager, udev, mounts the Pebble's Repository as part of the Urn's file system [71]. The control flow diagram of this P&U prototype is shown in Figure 43.

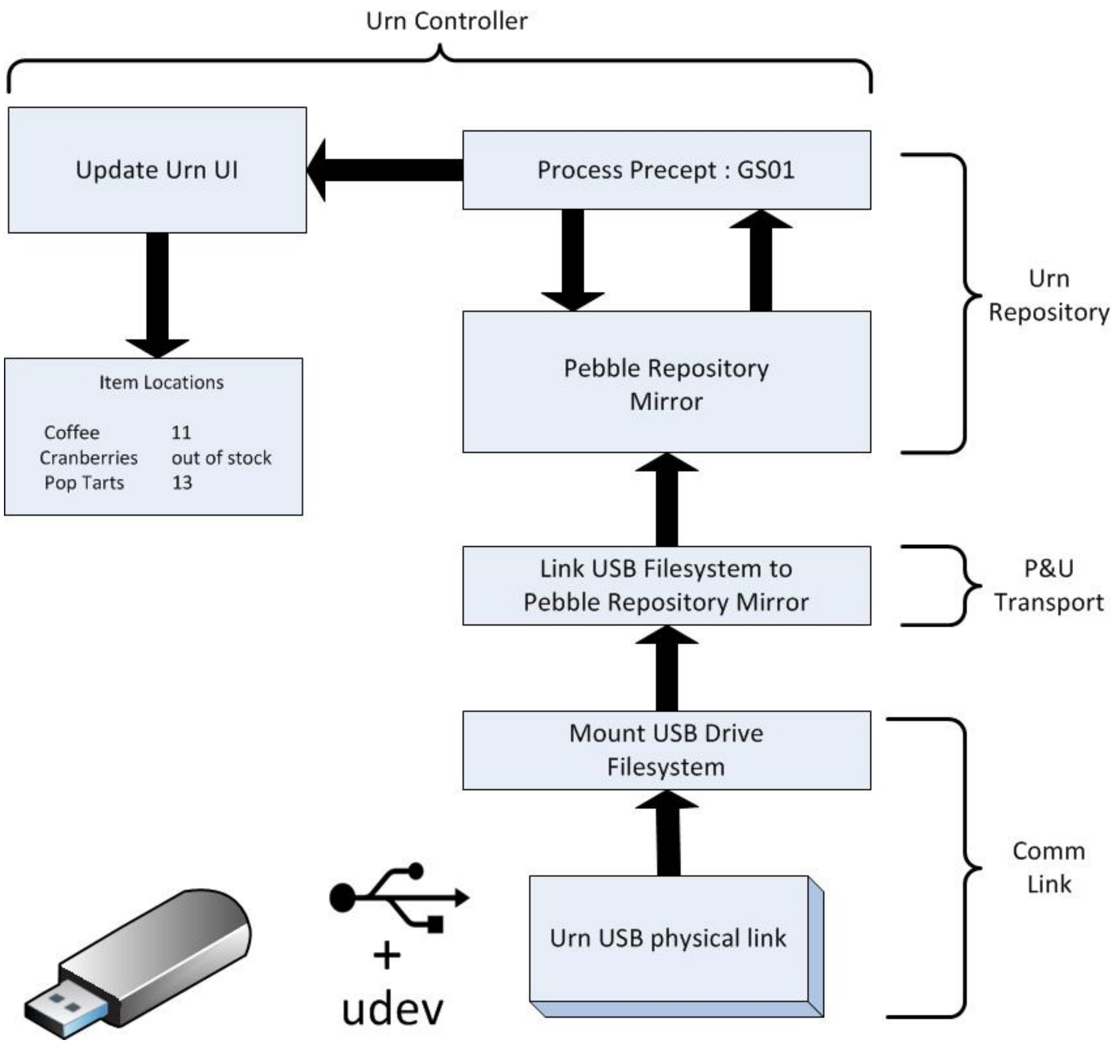


Figure 43 - Control Flow - Read/Write Pebble

The Pebble Repository of the Read/Write Pebble is illustrated in Figure 44. In addition to the data elements of the Read-Only Pebble Repository, the Read/Write Pebble Repository contains a user profile, user preferences, Remnants and an extended Precepts structure.

Definitions and use descriptions of these data elements are contained in Sections 4.3.2.2.1 and 4.3.2.3, respectively.

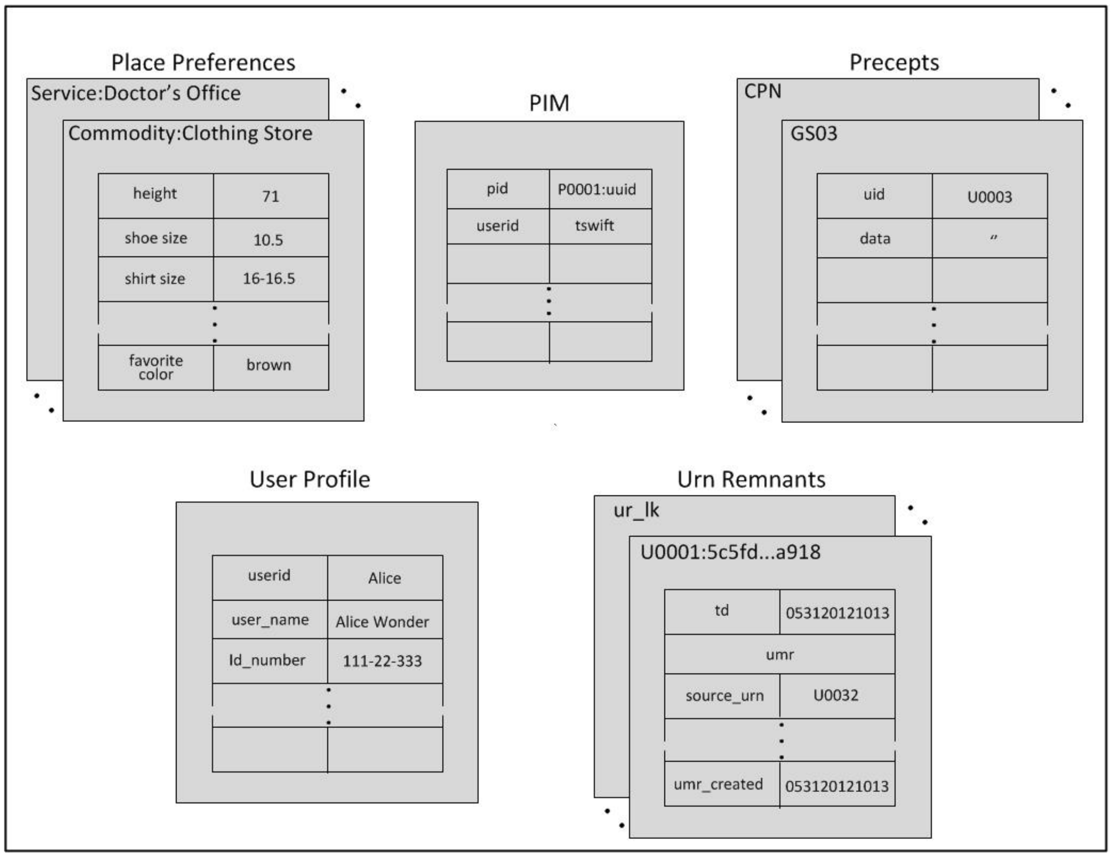
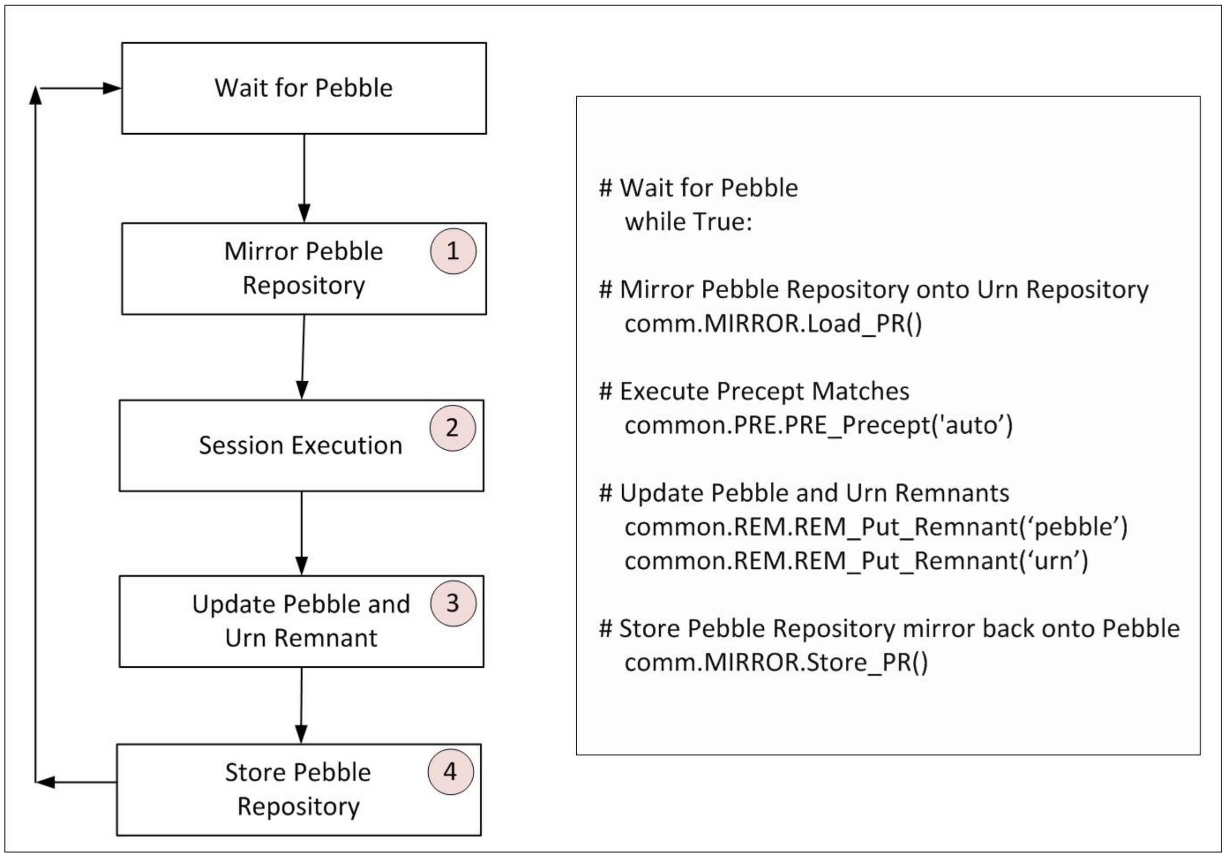


Figure 44 - Example of Read/Write Pebble Repository

Figure 45 shows the program flow and code for applications performing the interactions described in the Read/Write scenario in Section 5.4.1. Step 1 represents the Pebble Repository replication process where the Pebble Repository is transferred to the Urn Repository. P&U session execution occurs in Step 2. In the Read/Write Pebble scenario, Precept processing, Remnant queries and user preference and place profile comparisons are actions that occur during session execution. In Step 3, new Pebble and Urn Remnants are created containing snapshot data about this P&U session. The updated Pebble Repository is transferred from the Urn back to the Pebble in Step 4.



```
# Wait for Pebble
   while True:

# Mirror Pebble Repository onto Urn Repository
   comm.MIRROR.Load_PR()

# Execute Precept Matches
   common.PRE.PRE_Precept('auto')

# Update Pebble and Urn Remnants
   common.REM.REM_Put_Remnant('pebble')
   common.REM.REM_Put_Remnant('urn')

# Store Pebble Repository mirror back onto Pebble
   comm.MIRROR.Store_PR()
```

Figure 45 - Read/Write Program Flow and Code

5.4.3 Testing and Performance Profiling

This section describes testing and performance profiling of the Read/Write Pebble prototype. This study follows the rationale and procedures described in Section 5.3.3.

Figure 46 shows the P&U code blocks that are active during a P&U session with a Read/Write Pebble. This is the program structure of the P&U system depicted in the scenario presented in Section 5.4.1. The code block numbering scheme is defined in Table 23. Code block 1, WRAP_Comm, mounts the USB thumb drive's file system. Code block 2, MIRROR_Load_PR, links the mounted file system to the Urn Repository's PR_Mirror directory. Code block 3, PRE_Precept, executes any matching Precepts. Code block 4, REM_Put_Remnant, adds a Pebble Remnant to the Urn's Remnants data area. Code block 5, Mirror_Store_PR, updates the Pebble's Repository with the PR_Mirror contents.

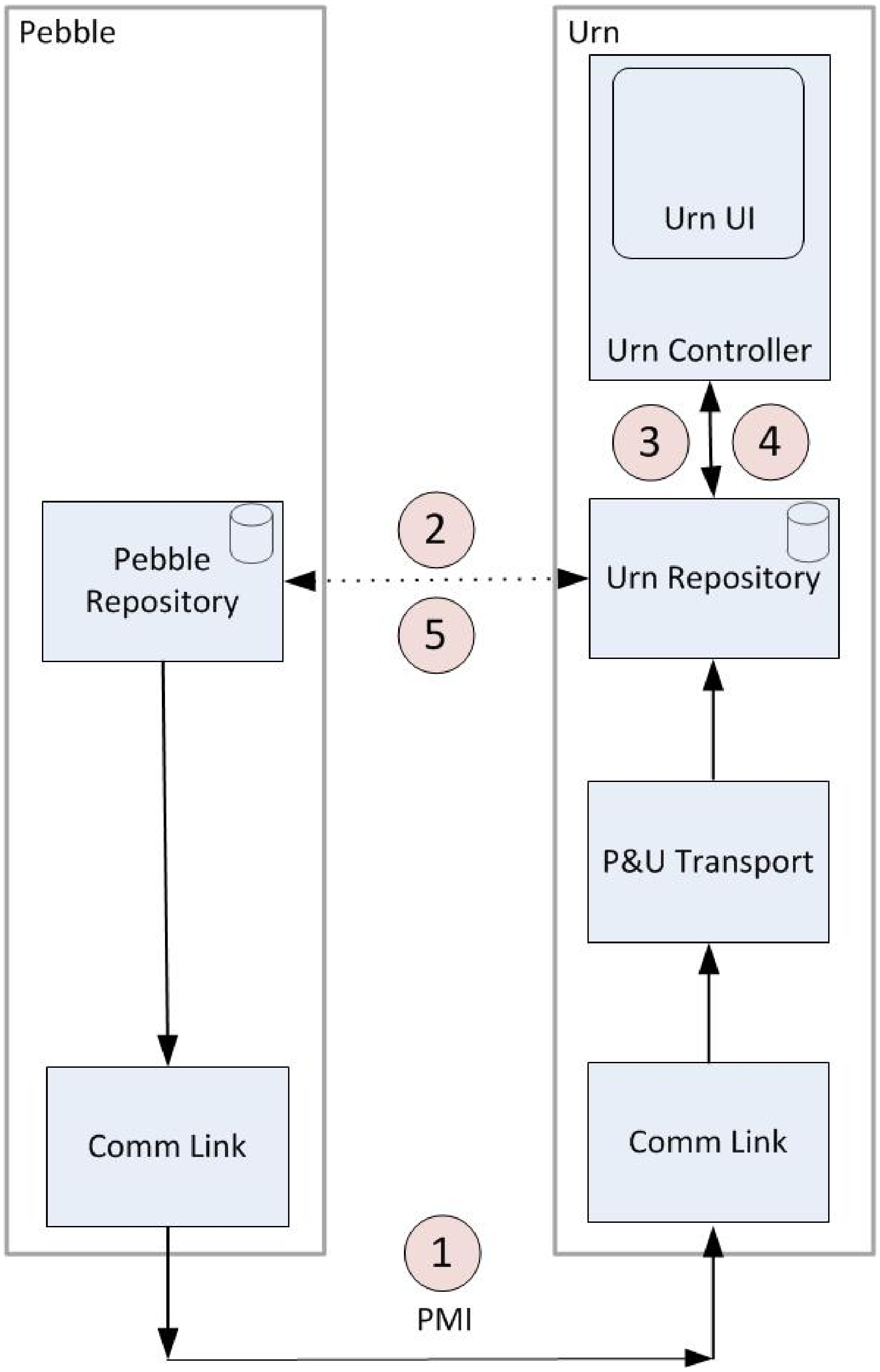


Figure 46 - Code Blocks of the Read/Write Pebble Instance

The P&U code blocks indicated in Figure 46 are analyzed using the profile module described in Section 5.3.3. Ten iterations of a P&U session using a Read/Write Pebble are analyzed. The Urn in this performance analysis exercise is the Grocery Urn. Table 26 contains the summarized information generated by this exercise.

Table 26 - Execution Times of the Read/Write Code Blocks

Generic	Read / Write	Average Execution Time (seconds)	Standard Deviation (seconds)
WRAP_Comm	Link_USB	0.133	0.005
Mirror_Load_PR	Load_USB_Mirror	0.018	0.006
PRE_Precept	PRE_Precept	0.490	0.007
REM_Put_Remnant	REM_Put_Remnant	0.021	0.002
Mirror_Store_PR	Store_USB_Mirror	0.057	0.002

The execution times are from code blocks run during the P&U session where Alice interacts with the Grocery Urn using a Read/Write Pebble based on a USB "thumbdrive". Alice initiates a P&U session by plugging her Pebble into the Grocery Urn.

Because of the Read/Write Pebble's increased storage and additional Repository elements, the P&U session processes additional code blocks. UMR_Message processing is now supported because the Pebble can now store Urn Remnants. The Mirror_Store_PR block detaches the replicated Pebble Repository at the end of the P&U session.

WRAP_Comm and Mirror_Load_PR code block execution times are shorter than those times reported for the Read_Only Pebble (Table 25). The USB Read/Write Pebble interacts with the Urn by means of a different physical interface. This interface and related differences in execution times are discussed in more detail in Section 5.6.

5.4.4 Outcomes and Discoveries

Outcomes observed and discoveries made during the implementation, testing and analysis of the P&U prototype and Read/Write Pebble are discussed in this section.

One observed outcome was the demonstration of increased functionality in the scenario attributed to the Read/Write Pebble's expanded, rewritable storage.

An aspect of the P&U framework design is the isolated physical communication layer. This design feature allows the changing of a P&U system's Pebble class interface by switching only the communication wrapper modules on the Urn. The Grocery Urn in this prototype interacts with a USB Read/Write Pebble.

The form factor of a Read/Write Pebble based on a USB thumb drive represents a tangible interface. The familiar USB interface is extremely easy to use. A P&U session is initiated by simply plugging the Pebble into the Urn; another example of an "Arrive and Operate" interface. The act of physically connecting a Read/Write Pebble to an Urn enforces physical proximity. A demonstration of near, situated, tangible interaction as discussed in Research Question 1.1 in Section 1.2.

5.5 P&U Prototype: Smart Pebble

This section describes a P&U prototype that uses a Smart Pebble. Pebbles in the Smart Pebble class have a Pebble Controller component. The Pebble Controller affords the Smart Pebble owner additional control over a P&U session. Figure 47 represents the architectural layers and components that support the Smart Pebble class.

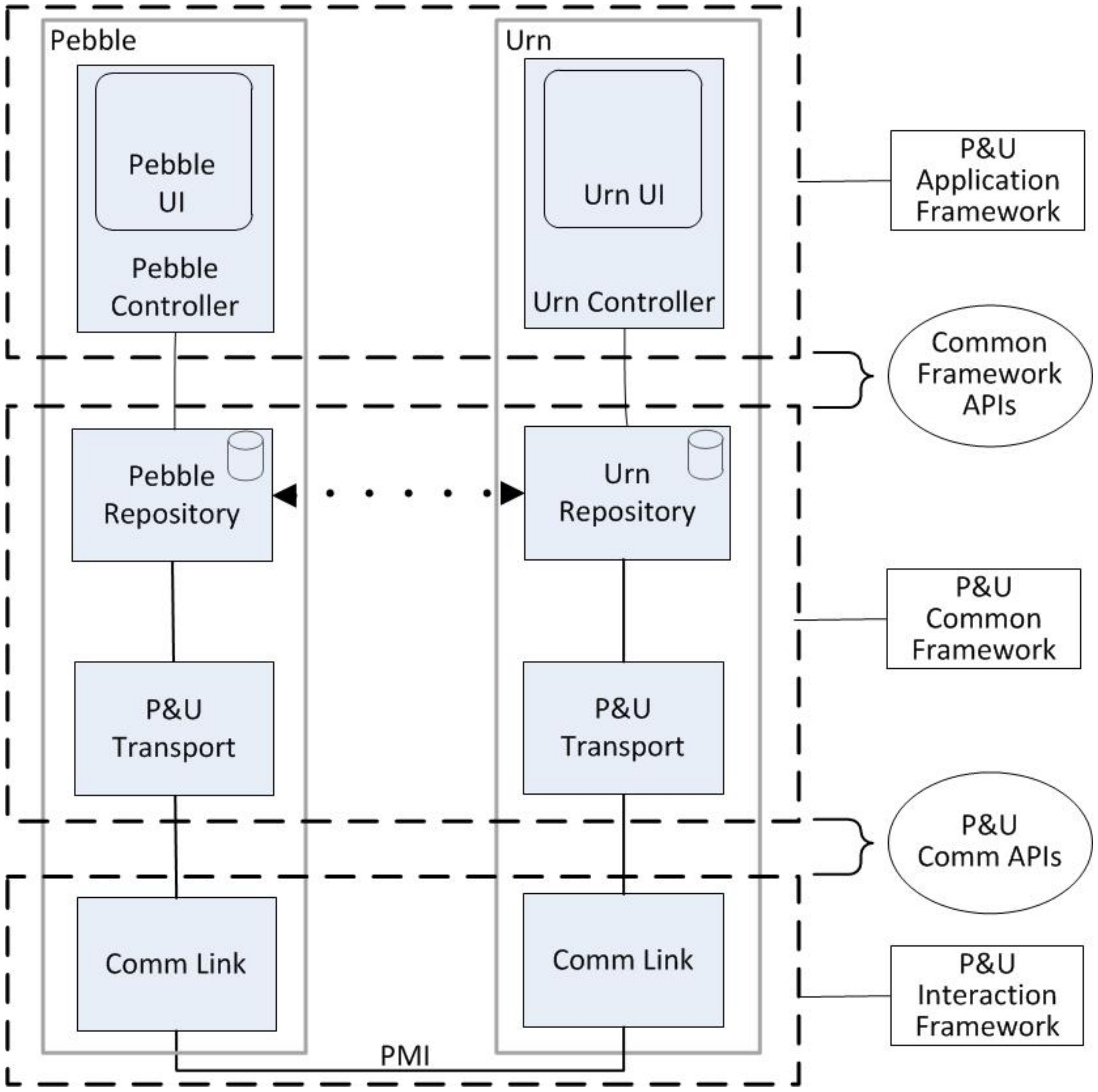


Figure 47 - P&U Framework - Smart Pebble

5.5.1 Scenario

The following scenario takes place in an environment that offers P&U systems supporting Smart Pebbles.

It's Wednesday evening, and Alice has just finished dinner. Now it's time to run an evening of errands. Alice makes sure she has her Pebble. Tonight is *class night, so she also takes her textbook off the nightstand as she leaves her apartment. Her first stop tonight is class.*

As Alice enters the classroom, she activates her Pebble. A list of roles appears on the Pebble's display. She selects the "Virginia Tech student" role and holds her Pebble next to the classroom Urn. The text "Assignment Accepted", "3F", and "Next Assignment Loaded" appear on the Urn's display. Alice acknowledges to herself that for this particular class she is in Group 3 assuming a role of facilitator. See also sees that her assignment due has been moved from her Pebble to the classroom Urn and her next assignment is now on her Pebble. Alice continues to the area of the classroom designated for Group 3 members. Class begins as the instructor gives assignments to the different groups. Class ends, and Alice heads to the gym.

Arriving at the multi-story exercise facility, Alice again activates her Pebble, and this time selects a role labeled "NewTown Fitness". She gains access by holding her Pebble next to the door receptacle of the gym Urn. The following message appears on the Urn's display: "1st floor treadmill available, 2nd floor elliptical available", "Current Resistance Level: 5". Her exercise routine consists of a sequence of exercises using different equipment located throughout the facility. The Urn has just displayed available equipment from her exercise profile. The Urn has also reminded Alice that her current resistance setting for the exercise equipment is level 5. Alice heads to the treadmill on 1st floor. Completing her exercise routine, Alice leaves for her last stop of the evening, the grocery store.

Alice enters the large FoodShop store with grocery list in hand. She activates her Pebble, selects a user role of "FoodShop grocery shopper", and holds the Pebble next to the grocery store Urn. The Urn acknowledges that it is her fifth visit this month and dispenses a 5% off coupon for this store purchase. The Urn also produces a printout showing each item on Alice's grocery list *associated with current store location and price. Looking over the printout, Alice sees that one item on her grocery list is out of stock, and also that the Grocery Urn has recommended a replacement for another item on her list. Aided by this printout, Alice begins her grocery shopping. Alice proceeds to the grocery store checkout already knowing the cost of her shopping trip. Alice gets in her car, turns on the radio, and heads home after another evening of errands. The reporter on the radio is discussing the growing concerns of identity theft. No worries here Alice thinks, knowing that she has control over her personal information and her Pebble. Now, my favorite TV show begins in 10 minutes.*

5.5.2 Application Considerations and Coding Details

The scenario in Section 5.5.1 illustrates the identity management support the Smart Pebble provides the Pebble owner during their interactions with different Urns. For each of the P&U sessions presented in the scenario, the Pebble owner selects the user role they wish to present to an Urn. This action controls the types of information transferred from the Smart Pebble to an Urn during a P&U session.

The Nokia N810 internet tablet is selected as the implementation platform for the Smart Pebble used in this prototype. This device was chosen because of its Unix/Linux operating system, Python interpreter and Bluetooth communication support. P&U framework API's, internal functions and tools run unmodified on the N810. The Bluetooth radio supports the physical communication link used in this prototype. Using the P&U framework, a Nokia N810 can perform as a Smart Pebble or an Urn, with neither implementation requiring any framework modifications. The Smart Pebble provides user feedback by means of a Pebble UI implemented on the N810's touchscreen display.

Prior to beginning a P&U session, the Pebble owner selects a particular role defining how they wish to be perceived by the Urn. Based on this role selection, the Pebble Controller modifies certain data elements in the Pebble's Repository. As an example, a role selection of "anonymous" would relocate Pebble Repository elements containing identifying user information to outside the Pebble Repository. This action prevents transfer of this information to the Urn during the Pebble Repository replication phase. A related consequence of role selection is the ability to configure a Pebble's Repository to better match a particular Urn. Selection of a "Virginia Tech student" role would alter Precepts, user profiles and preferences to align closer to services provided by the Classroom Urn.

In this prototype, the Urn advertises itself as a Bluetooth service. When the Smart Pebble discovers the Urn, a RFCOMM connection between the two devices is established and Pebble Repository replication begins. Pebble and Urn Bluetooth communication is implemented using Python's PyBluez library [72].

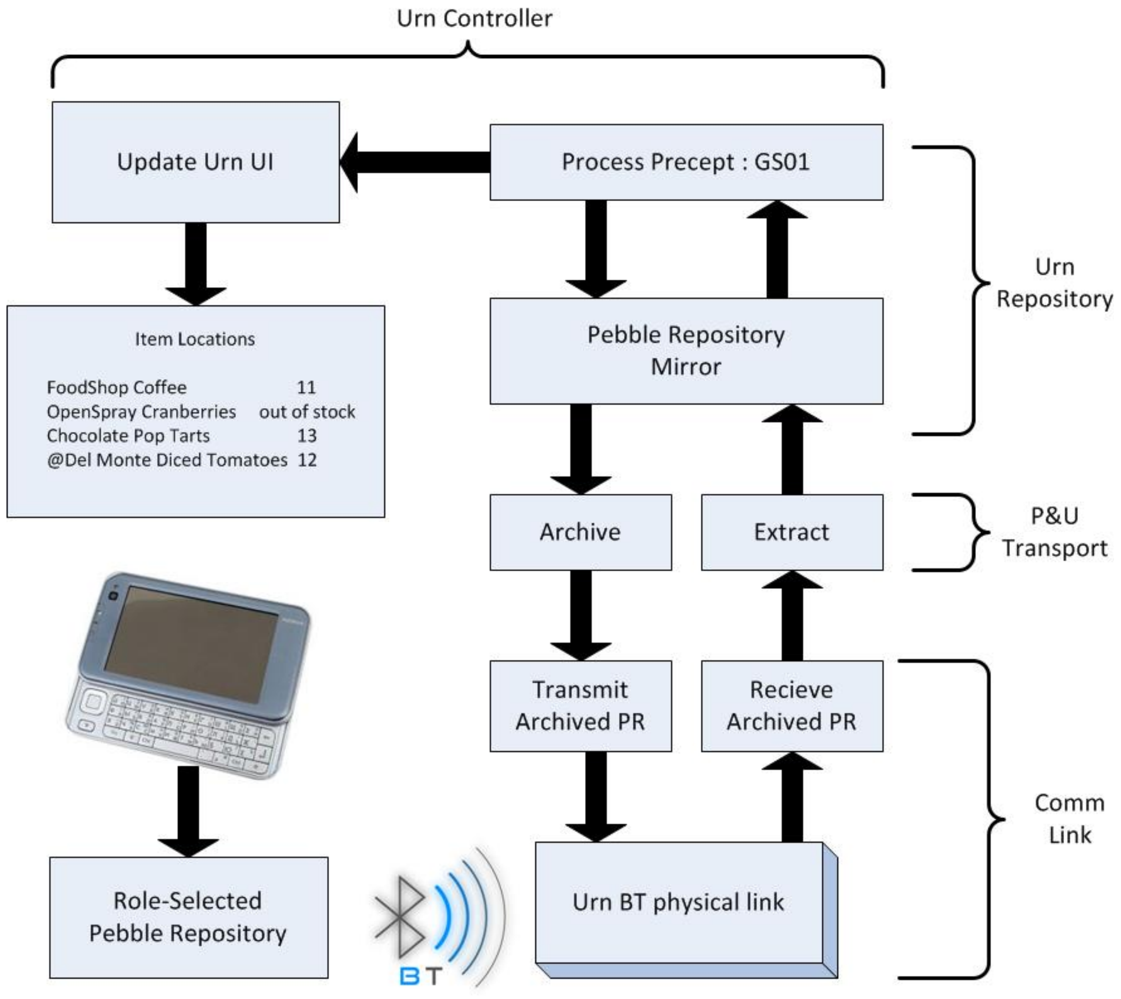


Figure 48 - Control Flow - Smart Pebble

The control flow diagram of this P&U prototype is shown in Figure 48. The Pebble Repository of the Smart Pebble is shown in Figure 49. Multiple user roles are implemented.

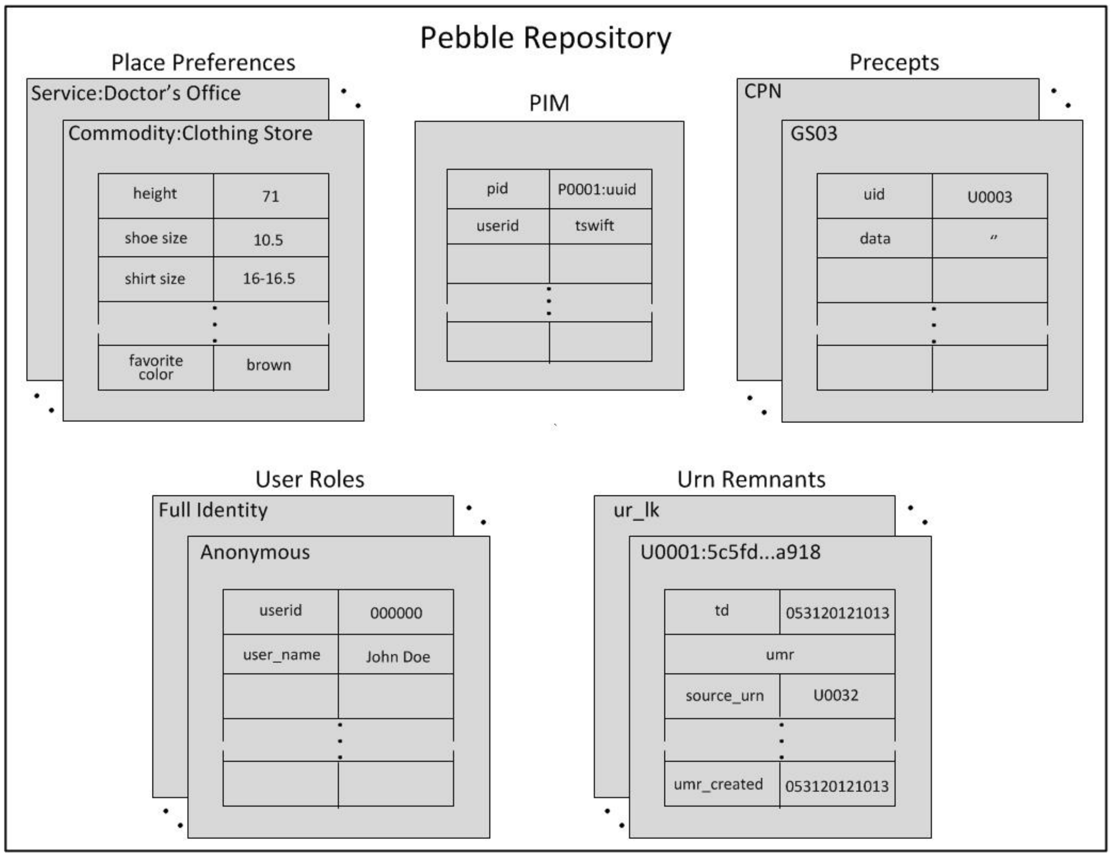
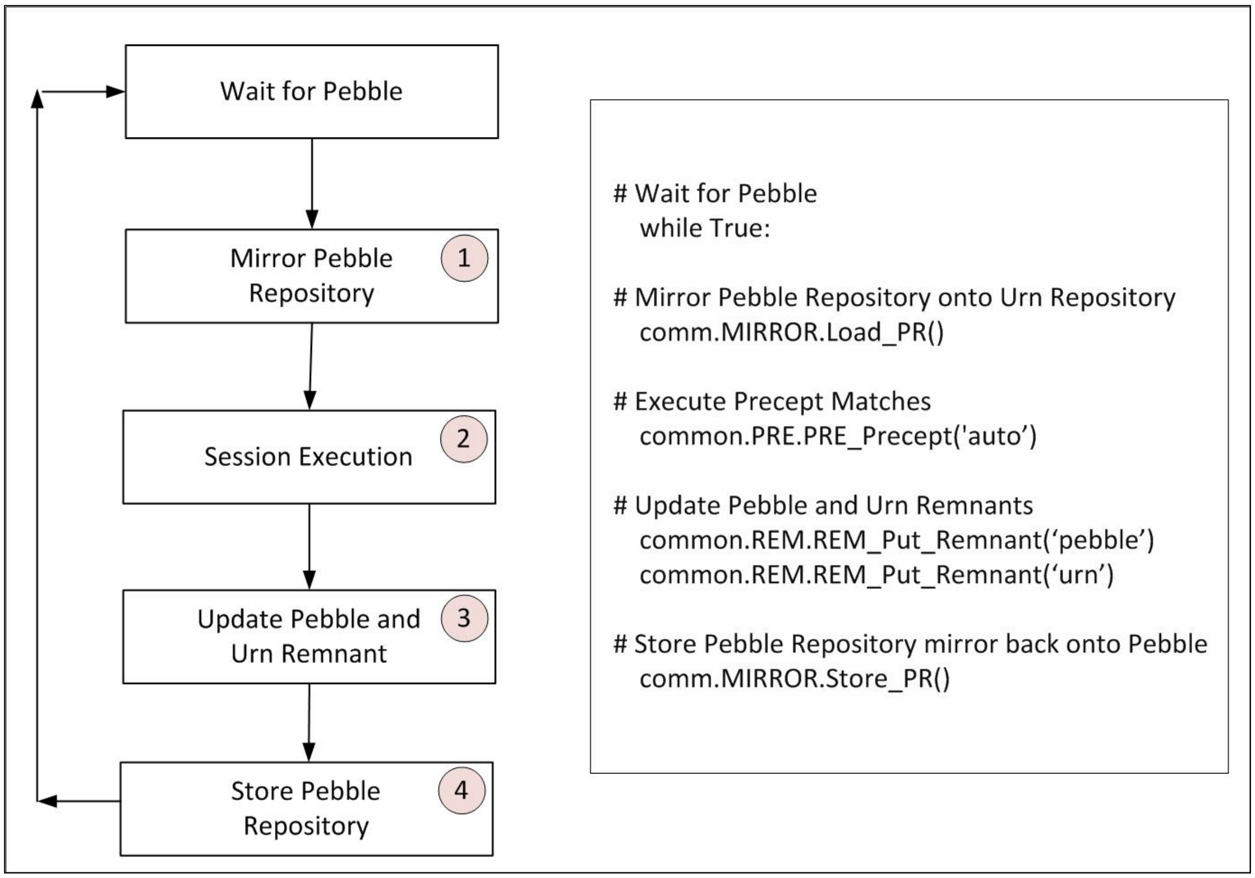


Figure 49 - Smart Pebble Repository

Figure 50 shows the program flow and code for applications performing the interactions described in the Smart Pebble scenario.



```
# Wait for Pebble
   while True:

# Mirror Pebble Repository onto Urn Repository
   comm.MIRROR.Load_PR()

# Execute Precept Matches
   common.PRE.PRE_Precept('auto')

# Update Pebble and Urn Remnants
   common.REM.REM_Put_Remnant('pebble')
   common.REM.REM_Put_Remnant('urn')

# Store Pebble Repository mirror back onto Pebble
   comm.MIRROR.Store_PR()
```

Figure 50 - Smart Program Flow and Code

5.5.3 Testing and Performance Profiling

This section describes testing and performance profiling of the Smart Pebble prototype. These exercises follow the rationale and procedures described in Section 5.3.3.

Figure 51 shows the P&U code blocks that are active during a P&U session with a Smart Pebble. The code block numbering scheme and descriptions are identical to those used with the Read/Write Pebble analysis presented in Section 5.4.3.

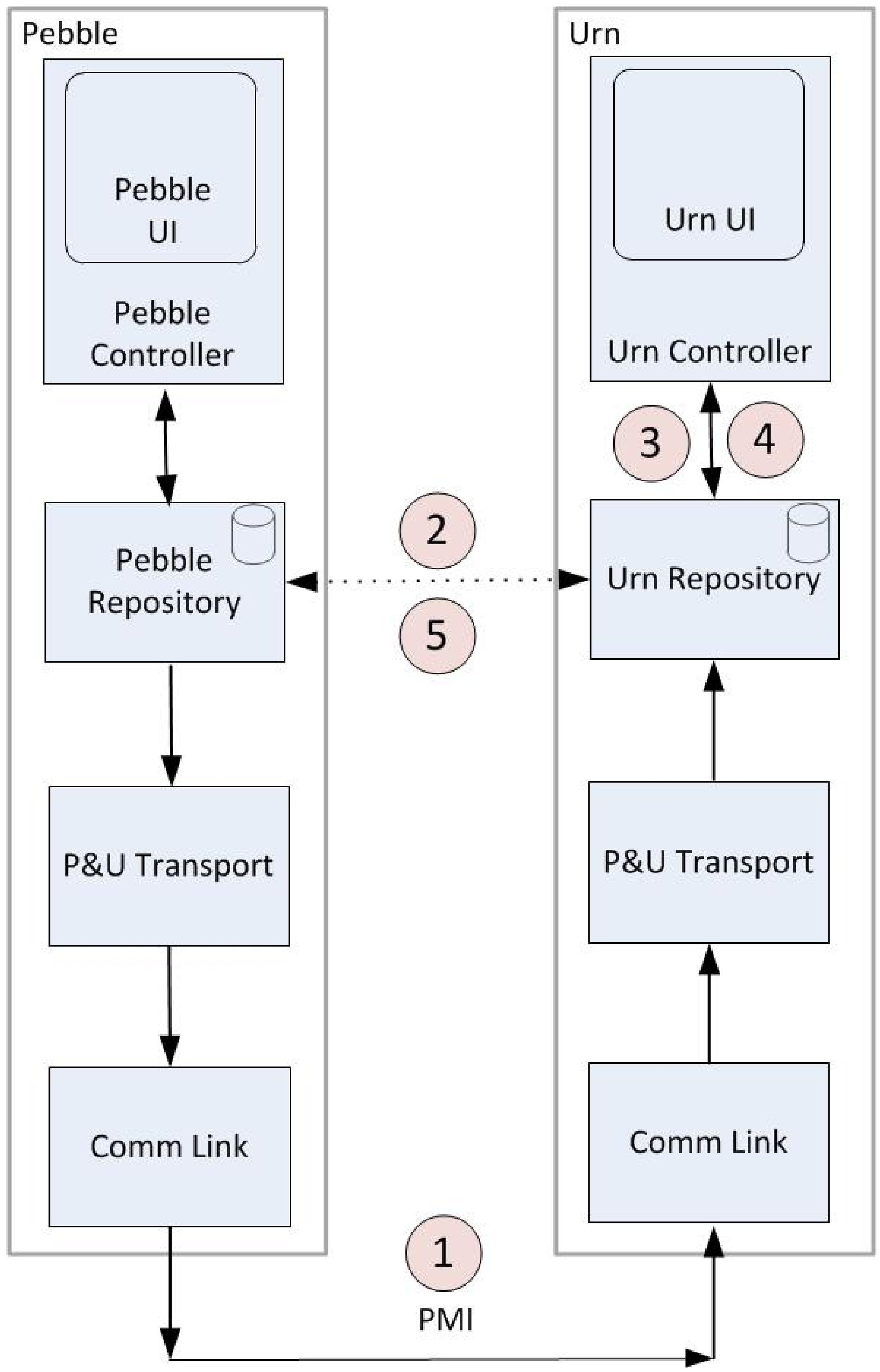


Figure 51 - Code Blocks of the Smart Pebble Instance

The P&U code blocks indicated in Figure 51 are analyzed using the profile module described in Section 5.3.3. Ten iterations of a P&U session using a Smart Pebble are analyzed. The Urn in this performance analysis exercise is the Grocery Urn. Table 27 contains the summarized information generated by this exercise.

Table 27 - Execution Times of the Smart Code Blocks

Generic	Smart	Average Execution Time (seconds)	Standard Deviation (seconds)
WRAP_Comm	Transfer_Bt	18.026	0.737
Mirror_Load_PR	Load_Bt_Mirror	0.032	0.003
PRE_Precept	PRE_Precept	0.510	0.008
REM_Put_Remnant	REM_Put_Remnant	0.038	0.002
Mirror_Store_PR	Store_Bt_Mirror	0.021	0.002

The execution times are from code blocks run during the P&U session where Alice interacts with the Grocery Urn using a Smart Pebble. Alice initiates a P&U session by selecting a role of “FoodShop Shopper” on her Smart Pebble. The Smart Pebble rearranges elements of its Repository to match the selected role, and initiates the Bluetooth client side application to connect with the Grocery Urn.

Incompatibilities between the Urn prototype platform (Dell Mini 8 laptop) and the Smart Pebble platform (Nokia N810) prevented the Bluetooth file transfer of the mirrored Pebble Repository from the Urn to the Pebble. Bluetooth file transfer does successfully work between two Nokia N810’s. In order to analyze this step, the Urn prototype is implemented on a second Nokia N810 using the P&U framework and profile tools. Table 28 contains the performance data for this analysis.

Table 28 - Execution Times of the Smart Code Blocks (N810)

Generic	Smart	Average Execution Time (seconds)	Standard Deviation (seconds)
WRAP_Comm	Transfer_Bt	.499	0.218
Mirror_Load_PR	Load_Bt_Mirror	0.182	0.026
PRE_Precept	PRE_Precept	1.15	0.148
REM_Put_Remnant	REM_Put_Remnant	0.193	0.022
Mirror_Store_PR	Store_Bt_Mirror	0.152	0.031
WRAP_Comm	Transfer_Bt	.397	.143

The WRAP_Comm code block's average execution time of 18 seconds is discussed in Section 5.6.

5.5.4 Outcomes and Discoveries

Outcomes observed and discoveries made during the implementation, testing and analysis of the P&U prototype and Smart Pebble are discussed in this section.

The role-selection process executed by the Pebble Controller performed as depicted in the scenario. Pebble Repository elements were successfully reorganized as defined by the selected user role.

The amount of time required to perform the Bluetooth service discovery step represents a P&U initialization time substantially longer than that of the Read-Only and Read/Write Pebbles. This observation warrants investigation of other wireless technologies for use in Smart Pebbles.

Plotting the Smart Pebble's WRAP_Comm time of 18 seconds against the other Pebble results dominates the graph as shown in Figure 52.

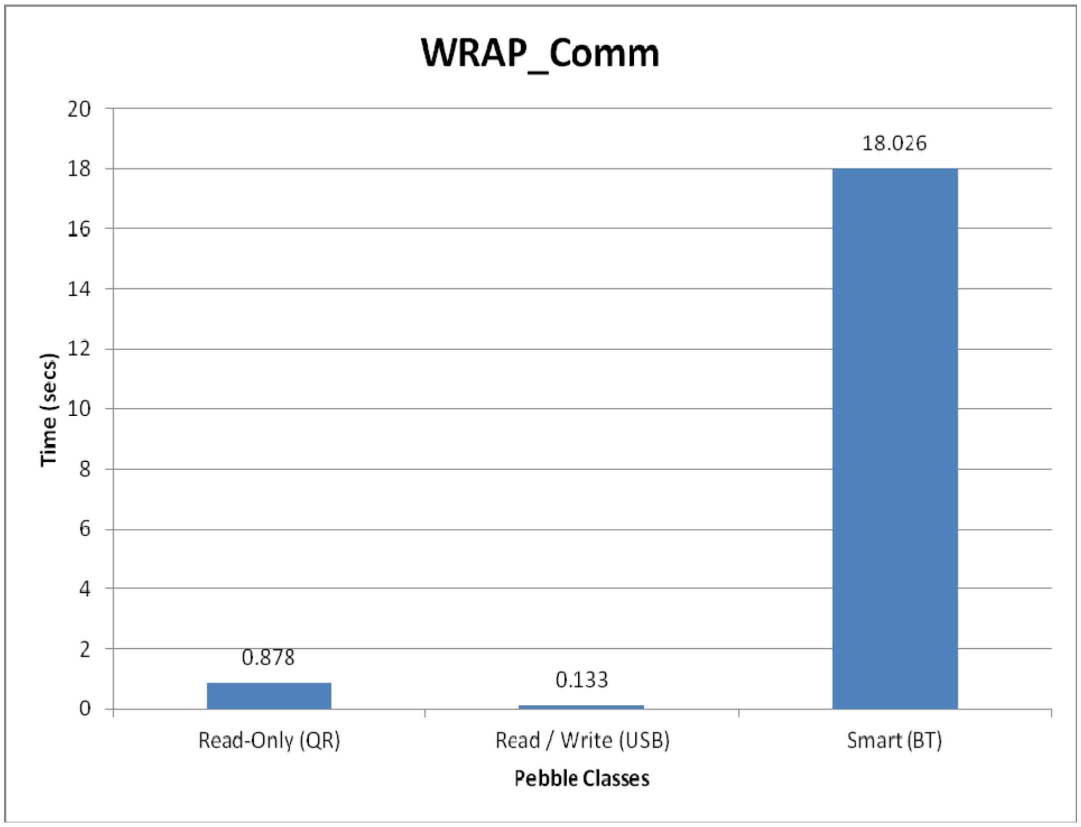


Figure 52 - WRAP_Comm Execution Times

The code performing the service discovery phase of the Smart Pebble prototype was removed from the process to obtain additional information from comparison plots of the different Pebble classes. Restated, for results presented below, performance results for the Smart Pebble's WRAP_Comm code block begin after the RFCOMM Accept() statement. Running the profile module against the Smart Pebble prototype now yields an average WRAP_Comm execution time of 0.256 seconds with a standard deviation of 0.018 seconds.

5.6 Performance Analysis

Performance data obtained by analyzing the basic P&U code blocks of the P&U sessions running on each of the three prototypes is represented in the following graphs.

The stacked column graph in Figure 53 shows the relationship of average code block execution times to the total execution time of a P&U session. The contribution of each execution time is compared to total execution times across the Pebble classes. Each column segment is labeled with the code block execution time it represents.

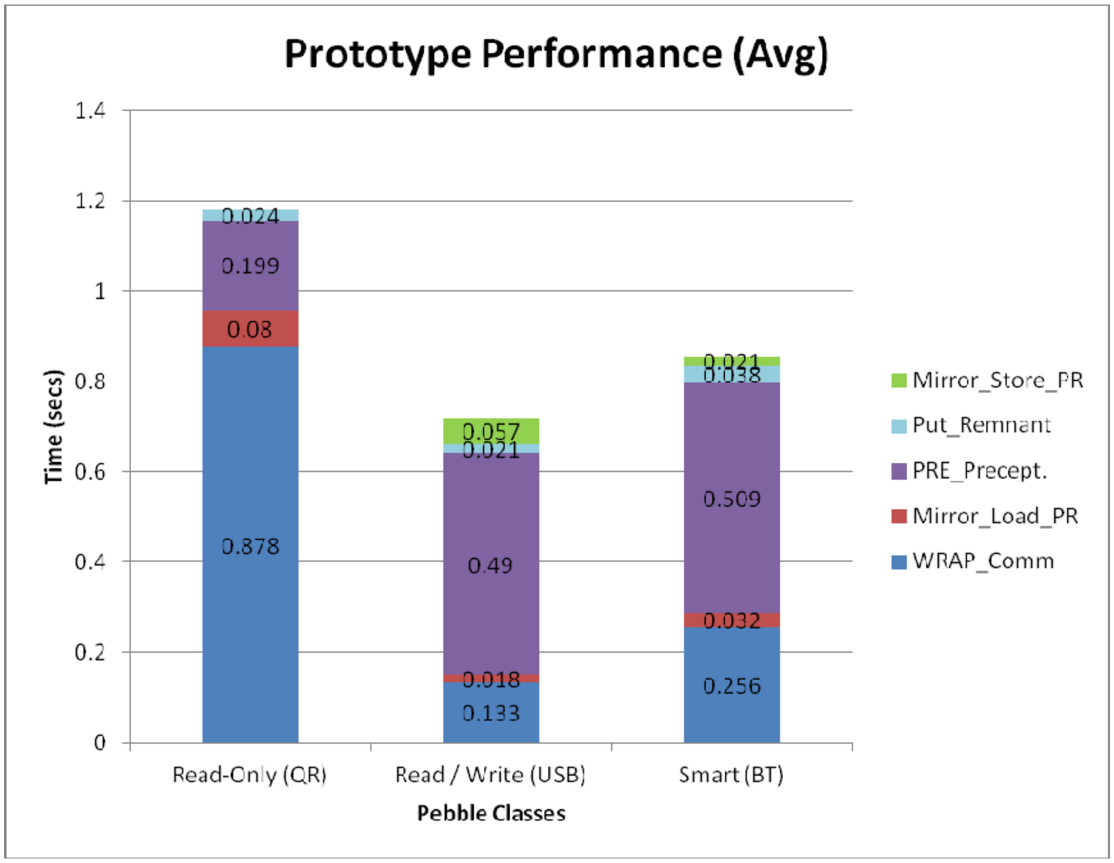


Figure 53 - Prototype Performance - Execution Times (Average)

Observations based on interpretation of this graph are as follows.

- The Read/Write Pebble based on the USB interface provides the fastest P&U session execution time because its WRAP_Comm execution time as shown on the graph is less than the WRAP_Comm execution times of the Read-Only or Smart Pebble. This is because the USB interface's WRAP_Comm code block uses file system mount and link operations that are faster than the Read-Only Pebble's QR decode operation or the Smart Pebble's file transfer operation.

- Precept processing times vary based on the nature and number of Precepts executed. The Urn Controller executes matching Pebble and Urn Precepts. The graph shows the Precept execution time of each Pebble class with the Grocery Urn. The Precepts offered by the Grocery Urn, as indicated in Table 1, are GS01, GS02 and GS03. The Read-Only Pebble, as shown in Figure 10, requests only the GS03 Precept. The GS03 Precept performs an Urn Remnant traversal to determine if the user qualifies to receive a store coupon. The Read/Write and Smart Pebbles with their larger storage capacity are configured to request all three grocery store Precepts offered by the Grocery Urn. Precept processing times are greater for the Read/Write and Smart Pebble than times for the Read-Only Pebble because of the additional Precepts executed. The graph also shows that the Precept processing times of the Read/Write and Smart Pebbles are equal. This is because their Precept requests are identical.

- Remnant processing times are similar across the three Pebble classes. In all three prototypes, the Put_Remnant method executed by the Urn Controller is performing the same operation.

The stacked column graph in Figure 54 represents the variability of execution times of the code blocks in the three prototypes.

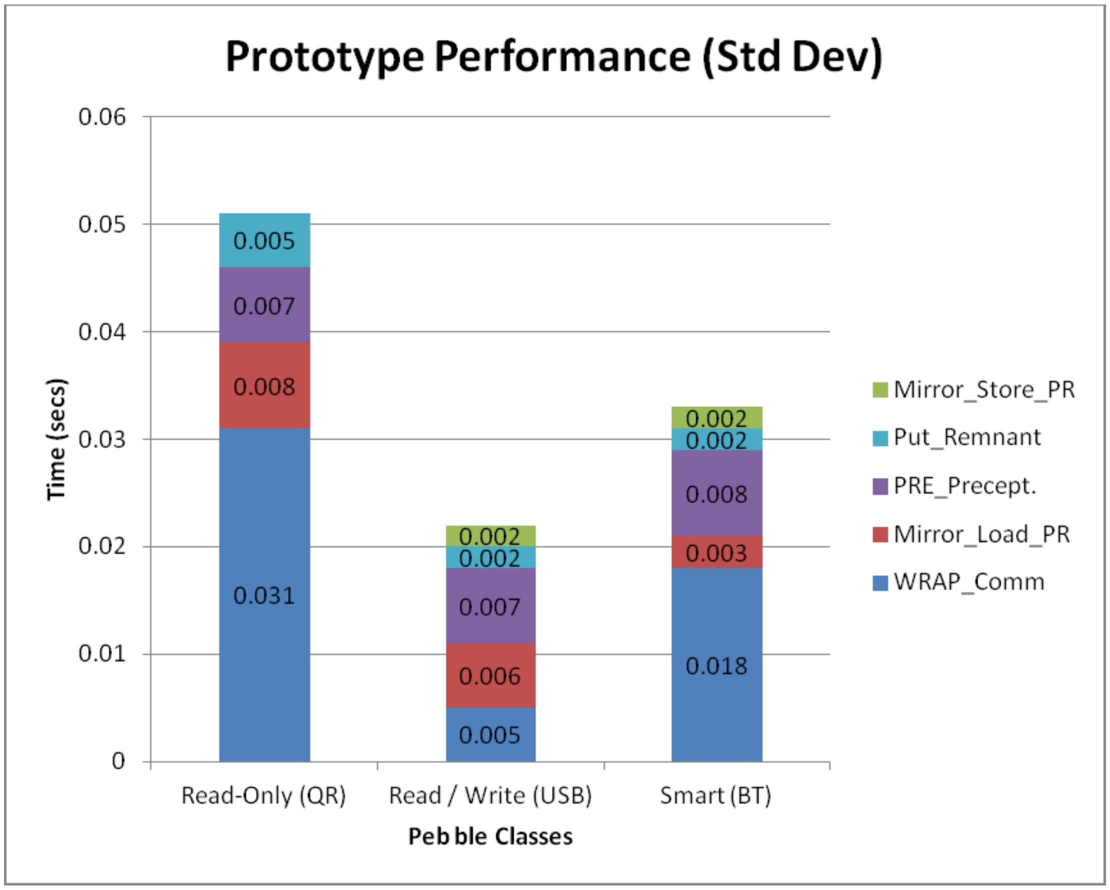


Figure 54 - Prototype Performance - Execution Times (Std Dev)

Observations are as follows.

- The greatest variability is shown in the Comm_WRAP code blocks of the Read-Only and Smart Pebbles. The Get_QR function of the Read-Only Pebble involves image processing steps and has the longest code block execution time of the prototypes. The Transfer Bt function of the Smart Pebble performs a file transfer of the Smart Pebble Repository across Bluetooth. The QR decoding and Bluetooth file transfer times are more likely to be affected by other operating system activities at runtime.

- Common framework operations like Precept processing and Remnant traversal demonstrate the same execution times with all Pebble classes over the ten profile run iterations. All P&U prototypes access and update the mirrored Pebble Repository

(PR_Mirror) through the P&U Common Framework. Also, Pebble and Urn Repository data structures are the same for all Pebble classes. Low execution time variability is the result of the same code executed against the same data storage on the same hardware.

Figure 55 contains a column graph comparing percentages that each code block execution time contributes to the total P&U session execution times.

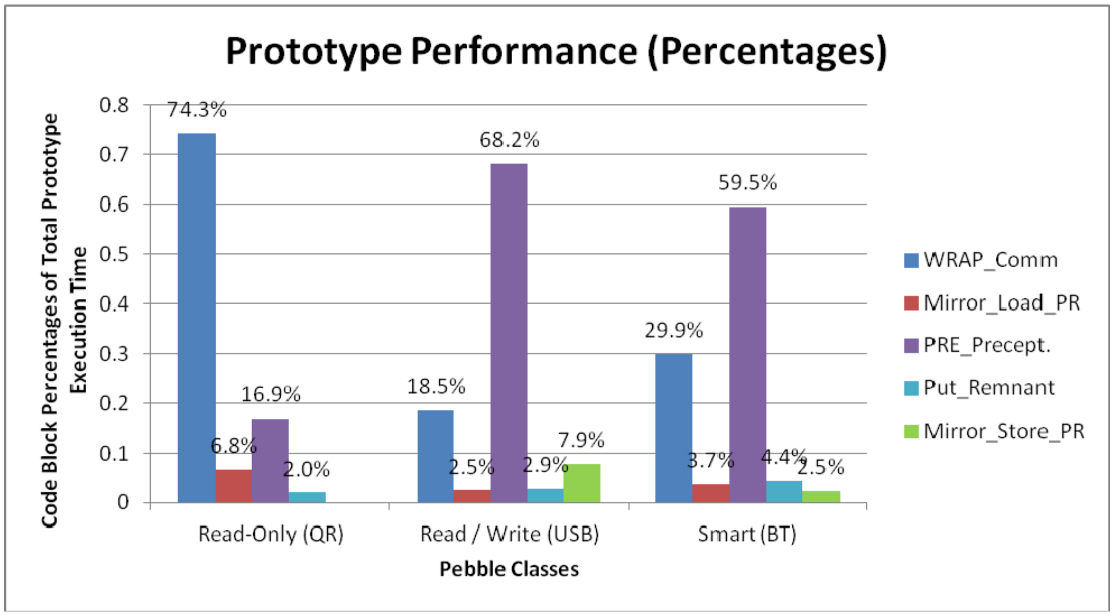


Figure 55 - Prototype Performance (Code Block % of Total Time)

Figure 56 is the performance graph of the Smart Pebble code blocks where both mirror Pebble Repository transfers are analyzed. File transfer times in each direction are similar.

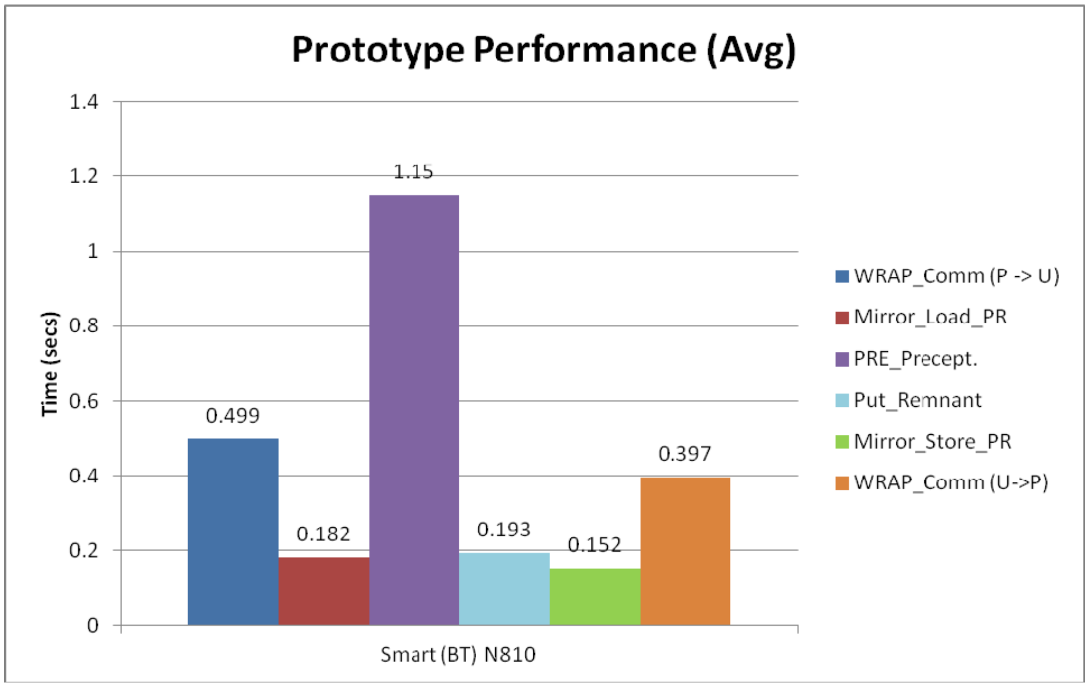


Figure 56 - Prototype Performance (includes Smart Pebble WRAP_Comm time)

Observations from this graph are as follows.

- The Read/Write USB-based Pebble prototype spent the least processing time (20% of total session time) establishing contact between Pebble and Urn. This is attributed to characteristics of the direct "wired" USB connection.

- 75% of the total session time for the Read-Write prototype session was spent in the PRE_Precept code block. This is attributed to fewer Precepts executed in the Read-Only session prototype. The Read/Write prototype session also demonstrated faster Pebble and Urn connect times than in the Read-Only prototype session.

- 74% of the total session time of the prototype using the Read-Only Pebble was spent in the WRAP_Comm code block. This is attributed to the execution times of the QR text decoding process in the Get_QR function.

- 97% of the Smart Pebble prototype's total execution time as shown in Figure 52 is spent in the WRAP_Comm code block performing Bluetooth service discovery. Bluetooth service discovery and connection times are acknowledged issues. To address this problem, researchers are studying ways to speed up the Bluetooth service discovery and connection phase [73], or couple Bluetooth implementations with other wireless technologies like IrDA [74].

Conclusions and observations regarding P&U performance based on this analysis are as follows.

- Initial tuning exercises on the P&U framework should focus on improving Pebble and Urn initial contact within the Comm WRAP layer. Comm_WRAP executions times represent 74% and 97% (Figure 56) of the total execution times of prototypes based on Read-Only and Smart Pebble instances respectively.

- The Read/Write Pebble based on a USB thumb drive offers the best performance in terms of execution time of the three Pebble instances tested. The Read/Write Pebble demonstrated the shortest Pebble to Urn connection and Repository transfer times.

- Stages of the all P&U prototype sessions using the Common Framework functions (PRE_Precept and Put_Remnant) demonstrated similar execution times. To restate, all P&U prototypes access and update the mirrored Pebble Repository (PR_Mirror) through the P&U Common Framework. Also, Pebble and Urn Repository data structures are the same for all Pebble classes.

- Not reflected in the graphs, the WRAP_Comm block in each of the prototypes successfully executed during all ten iterations of the testing. This observation demonstrates the stability of the P&U communication connections using the chosen PMI technologies.

5.7 Summary

This chapter describes the implementation of prototypes using the evolved P&U framework presented in Section 4.3. Experiences gained form this P&U instantiation work address the research questions established in Section 1.2. The following sections examine how these experiences answer the research questions.

5.7.1 P&U Interactions – Research Question 1.1

The performance analysis of the prototypes built on the P&U framework presented in Section 4.3.2 demonstrate stable and flexible communication connections between Urns and instances of the three Pebble classes. The communication technologies used in the framework's Comm layer enforced near, situated, tangible interactions between Pebble and Urn in all prototypes. Performance measurements of the code blocks managing the communication connections demonstrated consistent transfer times over multiple runs. Supporting the definition of interoperability presented in Section 4.1.1, the prototypes demonstrate the ability of an Urn to exchange information with different Pebble classes and to use the information that has been exchanged.

Tables 1 and 2 in Section 2.2.1 describe platform and architecture aspects of the three Pebble classes implemented in the P&U prototypes described in Section 5.2. Analysis of prototype performance discussed in Section 5.6 relates these aspects to the pros and cons of the physical interaction schemes used in each of the Pebble classes. The storage structure of the QR-based Read-Only Pebble is simple and constrained by storage size, limiting the functionality of P&U systems based on its use. The Smart Pebble's Bluetooth communication component with its multi-second service discovery delay fails to address the "Arrive and Operate" IoT challenge discussed in Section 1.1. The Read-Only and Read/Write Pebbles did not require their own power sources to interact with Urns.

5.7.2 P&U Opportunistic Elements – Research Question 1.2

The effects on user experiences by the prototypes demonstrate several opportunistic mechanisms provided by the P&U framework. Algorithms defining Remnant processing created an opportunity to encourage user participation by the award of a grocery store coupon. Operations on contextual information stored in a user profile Repository element facilitated the act of locating available exercise equipment in the gym. Rule sets controlling Precept matching enable services to manage homework assignments and effectively locate grocery items.

5.7.3 P&U Cyber Foraging – Research Question 1.3

Two of the three Pebble classes supported by the prototypes do not use power. The Read-Only Pebble instance based on a QR code does not perform computation, active communication, or user feedback. These functions are performed by the Urn. The same Urn relationship holds true for the Read/Write Pebble instance based on a USB "thumbdrive". The Smart Pebble instance based on an internet tablet requires power for computation

5.7.4 P&U Innovative Features – Research Question 2

Innovative features revealed through the design and implementation of the P&U prototypes include the following.

1. The P&U framework displays a "thin waist" when used in construction of Internet of Things and opportunistic computing systems. Sections 2.2.4 and 4.1.1 describe the "thin waist" or hourglass-shaped characteristic of architectures containing layered protocols.

2. The use of the Python language as a development tool aligned well with the requirements of P&U framework and prototype implementations.

   Python is an interpreted, object-oriented, high-level programming language [75]. Python is well-suited to rapid application development given its high-level built-in data structures, dynamic typing and dynamic binding. Supporting modularity and code reuse, Python serves as a "glue" language used to connect existing components.

Initial P&U research identified tuplespaces as the Pebble and Urn Repository datastore. Decisions on tuplespace implementations, language interfaces, and in some cases cost made a tuplespace selection difficult. Python's dictionary data structure provides tuplespace functionality including associative memories indexed by immutable-type keys. Core Repository elements and data operators are built using Python's dictionary data structure.

3. Repository data elements and associated operations can be combined programmatically to identify place-specific opportunistic situations. The following P&U code segment awards a discount coupon to the Pebble owner on an item identified as a specific personal preference if they have visited the store over ten times.

   Tom presents his Pebble to the store Urn as he enters his favorite clothing store. The store Urn presents him with a 15% discount coupon on all shoes in the store that are his size.

```
if count(Get_Remnant('pid')) >= 10:
  if preference_matchs = PRO_Comp() >= 1:
     item_preference_match = pick(preference_matchs)
     /* award discount on matched preference item */
```

5.7.5 P&U Problematic Features – Research Question 2

Problematic features revealed through the design and implementation of the P&U prototypes include the following.

1. The P&U session characteristic of the Urn waiting for Pebbles using a "block on wait" strategy is inefficient. A more efficient implementation would need to handle the various

P&U interface technologies. One approach is the modification of wrapper code to support event driven Pebble and Urn initialization.

2. P&U sessions do not support multiple Pebbles simultaneously. A session feature such as place adaptation to multiple Pebbles and their owners is not implemented. A framework modification to resolve this issue requires the addition of a Pebble registry sub-system that would track active Pebbles, a Pebble coordination data element in the Urn Repository, and a common framework function to map and interpret multiple user profiles.

3. While the use of Python is generally supportive of the implementation of Pebbles and Urns, there are caveats.

   Python's sophisticated data structures may make porting of the P&U framework to other languages and development environments difficult. Framework re-design may be required if different development and deployment environments are used. The degree to which P&U framework design was effectively isolated from its implementation is unknown at this time.

4. As described in Section 5.7.1, the prototypes focus on P&U support of the different Pebble classes. At this time, research has not extended Urn capabilities beyond the standalone Urns represented in the scenarios and prototypes. This area of research can be addressed as future work.

5. Earlier framework designs of the Comm and Transport layers did not provide an isolated physical communication layer. This issue has been resolved with the addition of the Pebble Repository replication step in the evolved P&U design, as presented in Section 4.3.2.2.2.

Chapter 6 concludes this dissertation by summarizing the research and identifying contributions. Possible future research directions are then discussed, followed by closing remarks.

Chapter 6: Conclusions

This chapter presents conclusions based on findings obtained from the exploration, design and implementation of a general Pebbles and Urns framework. Section 6.1 summarizes the body of work presented in this dissertation. This section restates the motivations for this research, presents the research questions derived from the motivations, and justifies the approach of this study to answer these research questions. Section 6.2 examines and interprets findings with respect to each of the research questions, relates the answers and findings of the study back to the conceptual framework upon which the study was designed and discusses additional issues raised during the course of this research. Section 6.3 identifies contributions of this research. Section 6.4 presents recommendations and directions for potential future research.

6.1 Research Summary

The following sections summarize the motivation for and approach taken by this research.

6.1.1 Research Motivation

The exploration of Pebbles and Urns takes place within the context of the Internet of Things and opportunistic computing. As presented in Section 1.1, prior research in these two areas of pervasive computing identified challenges associated with their underlying technologies [3, 5, 6]. These are listed and discussed below.

1. Decision support – IoT systems need to support autonomous decision making to reduce everyday decision making tasks.
2. New device interfaces – The study of new human-to-device and device-to-device interfaces should consider simple, dedicated personal devices that are easier to operate than smartphones.
3. Deployment – Architectures and frameworks are needed to support large-scale IoT and OC system deployments.

4. Scalability – Communication and service discovery components of the IoT need to function in large-scale environments.
5. “Arrive and Operate” – Interactions between humans and devices in the IoT require spontaneous connection and organization.
6. Interoperability – The diversity of many different types of smart objects require common practices and standards.
7. Discovery – Services for “things” must be automatically identified.
8. Software complexity – Software infrastructures are needed to manage smart objects and provided services.
9. Data volumes – “Real-world” awareness scenarios require volumes of local and remote data.
10. Data interpretation – The interpretation of local context to trigger further action is not trivial.
11. Security and personal privacy –Personal privacy within the IoT requires support of selective access control of services and information among users and the many smart objects.
12. Power supply –The many devices in the IoT need to take advantage of energy saving advances in hardware and software.
13. Interaction and short-range communication – Development of new interaction techniques and processes between humans and devices in the IoT is needed.
14. Connectivity – Ways to manage the stability, duration, and frequency of OC devices must be studied.
15. Incentives – There is a need to create reasons for initiating OC device conversations.
16. Context management – There is a need for OC systems to manage context representing user activity within a place.

These challenges serve as motivation for the design and implementation of the P&U framework. As a contribution to IoT and OC research domains, P&U is proposed to better understand and address some, but not all, of the key needs described above.

6.1.2 Research Questions

Research questions derived from the IoT and OC challenges discussed in Section 1.1 and restated in Section 6.1.1 define and drive the design, implementation, and analysis of the P&U framework. Challenges contributing to RQ 1.1, RQ 1.2 and RQ 1.3 are denoted in brackets (<>).

RQ 1. How would properties and characteristics of the P&U architecture advance the design and deployment of IoT and opportunistic computing implementations? Guiding components of this research question include the following.

RQ 1.1 <2, 5, 6, 8, 13, 14> What interoperability and interaction features and characteristics of the P&U architecture provide a stable and flexible communication connection supporting near, situated, tangible interaction between devices participating in the application domain of opportunistic computing?

RQ 1.2 <1, 7, 8, 9, 10, 11, 15, 16> Within the P&U architecture, what protocols, rule sets, algorithms, abstractions, and context management identify a relevant service domain for parties participating in a P&U-based opportunistic computing session?

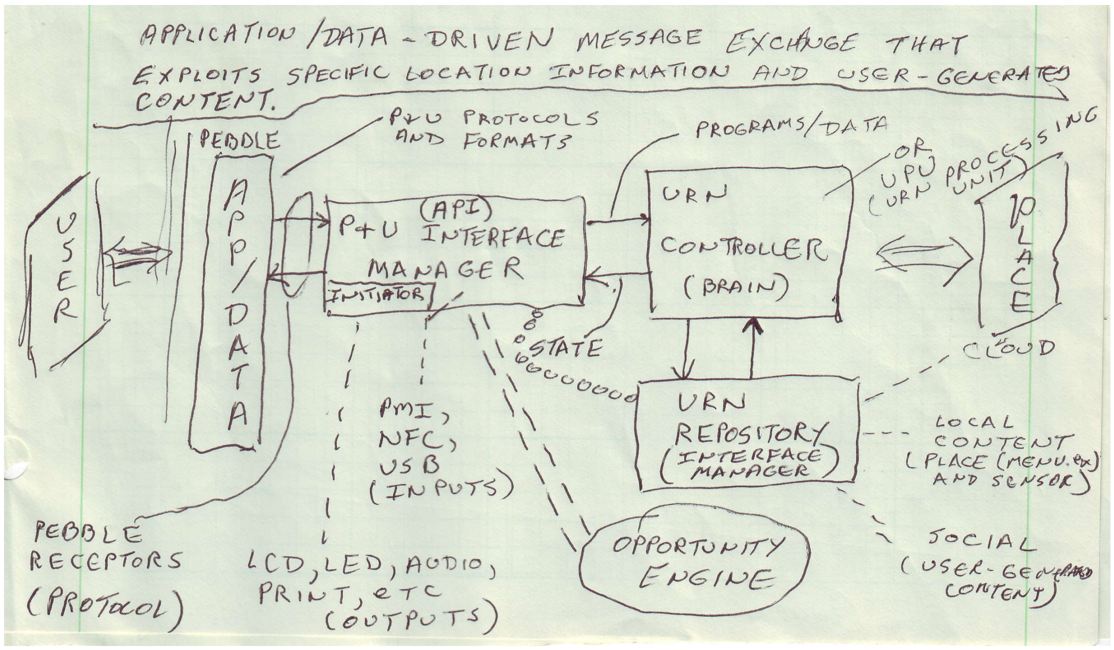
RQ 1.3 <2, 8, 12, 13> How can P&U designs utilize cyber foraging strategies such as power management and cloud services?

RQ 2. What innovative, or problematic, features are revealed through the design and implementation of a P&U instance and, for problematic features, can the problem areas be addressed?

6.1.3 Research Approach

Serving as a summary of this dissertation, this section describes the approach taken in the research of Pebbles and Urns. Topics include the concept of Pebbles and Urns, the study of prior related work, the initial and evolved design of a general P&U framework, and the implementation, testing and analysis of P&U prototypes.

The research motivation and derived research questions discussed in Sections 6.1.1 and 6.1.2 respectively, guided initial P&U concept design. Figure 57 shows a sketch of an early P&U concept. This early concept evolved and was refined by investigating related research areas and by exploring the use of P&U for application scenarios.



APPLICATION/DATA - DRIVEN MESSAGE EXCHANGE THAT EXPLOITS SPECIFIC LOCATION INFORMATION AND USER-GENERATED CONTENT.
PEBBLE
P+U PROTOCOLS AND FORMATS
PROGRAMS/DATA
OR UPU (URN PROCESSING UNIT)
USER
APP/DATA
P+U (API) INTERFACE MANAGER
INITIATOR
URN CONTROLLER (BRAIN)
PLACE
STATE
CLOUD
URN REPOSITORY (INTERFACE MANAGER)
LOCAL CONTENT (PLACE (MENU, etc) AND SENSOR)
PMI, NFC, USB (INPUTS)
LCD, LED, AUDIO, PRINT, etc (OUTPUTS)
OPPORTUNITY ENGINE
SOCIAL (USER-GENERATED CONTENT)
PEBBLE RECEPTORS (PROTOCOL)

Figure 57 - Early P&U Concept Sketch

Core elements of the P&U architecture, Pebble, Urn, and Information Repository, are introduced in Chapter 2. Attention to the research questions focused design efforts on P&U interactions, services, and cyber foraging. Use cases were written to demonstrate and better understand the application of P&U to everyday settings. A review of current research activities relevant to the P&U architecture is discussed in Chapter 3. Topics include mobile human and device interactions, information management of user-generated content, service discovery and cyber foraging.

Information obtained by this research helped refine the P&U concepts and architecture introduced in Chapter 2. Addressing the research questions in Section 1.2 with respect to the P&U concepts produced an initial P&U architecture described in Section 4.2. An initial P&U framework was implemented based on the components and connectors of the initial architecture. A simple prototype was constructed using this framework. The prototype successfully demonstrated a near, tangible interface and an isolated physical communication layer.

Examination of the simple prototype with respect to the research questions discussed in Section 1.2 revealed several open issues with the initial P&U design. A review of current work in the areas of service composition, service delivery and knowledge management identified the need to enhance the initial P&U model's composition and management of Repository contents and context. Repository enhancements included the addition of data elements supporting a simple service request process and a caching mechanism that collects P&U session information. New APIs and internal functions required to manage the new Repository data elements were designed and implemented. These additions addressed Research Question 1.2 by defining elements of a P&U service domain that assisted with the identification of opportunistic situations. To further study P&U interactions noted in Research Question 1.1, the isolated communication layer of the initial architecture was extended to support different Pebble classes. Pebble class support required a re-design of the P&U Comm layer in the initial framework. The layer, renamed as the P&U Transport layer, is responsible for the Pebble replication process described in Section 4.3.2.2.2.

Three Pebble classes were characterized in Section 2.2.1, a Read-Only Pebble, a Read/Write Pebble, and a Smart Pebble. Introduction of the Smart Pebble expanded the P&U framework as discussed in Section 5.5. The addition of a Pebble Controller provided increased capabilities including computation, active communication and Repository management. A realization during the course of this work is that, while all Pebble classes enable the goal of near, situated, tangible interaction, the configuration and structure of the different Pebble classes defines the specific capabilities of the P&U system with which they are used.

Section 5.2 describes the validation process for the P&U framework. Prototypes of P&U systems interacting with Pebble instances of the three Pebble classes were built using the framework. The prototypes demonstrate system behaviors that address the research questions. Sections 5.3.3, 5.4.3 and 5.5.3 analyze the performance of the three prototypes. Results are presented in Section 5.6.

6.2 Research Questions and Answers

The following sections apply the observations and findings obtained from the design, implementation and analysis of the P&U prototypes described in Chapter 5 to the research questions presented in Section 1.2. Each research question is restated followed by discussion.

6.2.1 Research Question 1

How would properties and characteristics of the P&U architecture advance the design and deployment of IoT and opportunistic computing implementations?

As discussed below in Section 6.2.1.1, the P&U framework provides a stable and flexible communication connection supporting near, situated, tangible interactions.

As discussed in Section 6.2.1.2, abstractions and contextual elements of the P&U Repositories define a relevant service domain for parties participating in a P&U-based opportunistic computing session.

As discussed in Section 6.2.1.3, the P&U framework design demonstrates cyber foraging strategies.

This P&U framework represents an instance of the components and connectors of the P&U architecture. The Interaction Framework defines a standard communication mechanism using extendable wrapper code modules to link Pebbles and Urns. Repositories in the Common Framework layer contain well-defined contextual data structures. The Application Framework uses an API to access and manipulate Repository data elements with specific operations designed to identify opportunistic situations. These P&U features contribute to the definition of a P&U design pattern, discussed in Section 2.2.4. Functioning as a common vocabulary used by software developers to build P&U systems, the P&U framework can serve as a design pattern to advance the design and deployment of IoT and opportunistic computing implementations.

6.2.1.1 Research Question 1.1

What interoperability and interaction features and characteristics of the P&U architecture provide a stable and flexible communication connection supporting near, situated, tangible interaction between devices participating in the application domain of opportunistic computing?

The near, situated, tangible aspects of the interaction between the Pebble and the Urn are defining characteristic of the P&U system. These characteristics are enforced by the P&U Interaction Framework described in Section 4.3.2.1 and its use of the physical mobile interaction (PMI) technologies described in Section 2.3 and 3.1.3.

The P&U framework design based on the layered architecture discussed in Sections 4.3.2 and 4.3.2.1 supports a flexible communication connection by means of interchangeable communication wrappers.

Testing of the P&U prototypes described in Section 5.6 demonstrated stability of the P&U communication connection. There were no failures of P&U communication connections during execution of the three prototypes. The timing of data transfer activities of a P&U session contribute to communication connection stability. There are two periods of communication activity during a P&U session. The first is the transfer of the Pebble Repository to the Urn at the start of the P&U session. The second is the transfer of the Pebble Repository back to the Pebble from the Urn. No communication occurs between the Pebble and the Urn at other times during the P&U session.

6.2.1.2 Research Question 1.2

Within the P&U architecture, what protocols, rule sets, algorithms, abstractions, and context management identify a relevant service domain for parties participating in a P&U-based opportunistic computing session?

The following abstractions and contextual elements define a relevant service domain for parties participating in a P&U-based opportunistic computing session.

Remnants are the "digital footprints" and "information scraps," described in Section 3.2.3, of a P&U system. Remnant traversal, discussed in Section 4.3.2.3, provides a view of past Pebble and Urn interactions stamped with user, time and location information. This information can be used to generate opportunistic situations as described in Section 5.7.4. Performing analysis of Remnant data in a P&U system could assist areas like social behavior research and marketing and sales planning.

User profiles and place preferences are used by the P&U system to provide individualized services. Profile evaluation is described in Section 4.3.2.3.

Precepts represent Pebble service requests and Urn provided services. Precept matching is discussed in Section 4.3.2.3.

Urn Message Relay described in Section 4.3.2.3 is a mechanism that transports messages residing on Pebbles from one Urn to another Urn. Messages could serve as an event trigger activating a particular service on the user's behalf.

As described in Section 5.7.4, these contextual data elements and associated operations can be combined programmatically to identify place-specific opportunistic situations.

6.2.1.3 Research Question 1.3

How can P&U designs utilize cyber foraging strategies such as power management and cloud services?

As discussed in Section 3.4, cyber foraging is the opportunistic use of nearby computing resources by small mobile devices. Mobile devices can achieve faster compute performance, access larger datastores, and conserve power by offloading tasks to more capable, nearby surrogate computers.

Cyber foraging is opportunistic in that mobile devices take advantage of each other's resources during random encounters. Pebble and Urn encounters are less random given the Urn's fixed location associated with the place it represents, and the Pebble owner's sense of purpose in visiting that place. Nevertheless, P&U design utilizes cyber foraging strategies.

As described in Section 5.7.3, two of the three Pebble class instances, the Read-Only Pebble and Read/Write Pebble, do not perform computation, active communication or user feedback activities. They do not require power. These Pebbles demonstrate a passive interaction role with the Urn assuming the cyber foraging responsibility of offloading Pebble information by means of the Pebble Repository replication process described in Section 4.3.2.2.2. The Urn performs computation on the offloaded Pebble information and provides user feedback through the Urn UI. Like the cloudlet described in Section 3.4, the Urn represents the surrogate computer in the P&U session.

Both the cloudlet infrastructure and the three-tier cyber foraging system described in Section 3.4 facilitate cloud access by means of an intermediary device. The P&U design presented in Sections 4.2.1 and 5.1 supports this cyber foraging concept with the Urn serving as the intermediary device. P&U cloud access is a function of the Urn Repository. This functionality is an area for potential future work.

6.2.2 Research Question 2

What innovative, or problematic, features are revealed through the design and implementation of a P&U instance and, for problematic features, can the problem areas be addressed?

Section 5.7.4 identified the following innovative features revealed through the design and implementation of the P&U prototypes.

1. The P&U framework uses a layered structure that displays a "thin waist" when used in construction of Internet of Things and opportunistic computing systems. Sections 2.2.4 and 4.1.1 describe the "thin waist" or hourglass-shaped characteristic of architectures containing layered protocols. This "thin waist" allows the Pebble and Urn application functionality and the communication functionality to be decoupled, thus simplifying system design and implementation.

2. The use of the Python language as a development tool aligned well with the requirements of P&U framework and prototype implementations.

   Python is an interpreted, object-oriented, high-level programming language [75]. Python is well-suited to rapid application development given its high-level built-in data structures, dynamic typing and dynamic binding. Supporting modularity and code reuse, Python serves as a "glue" language used to connect existing components.

   Initial P&U research identified tuplespaces as the Pebble and Urn Repository datastore. Decisions on tuplespace implementations, language interfaces, and in some cases cost made a tuplespace selection difficult. Python's dictionary data structure provides tuplespace functionality, including associative memories indexed by immutable-type keys. Core Repository elements and data operators are built using Python's dictionary data structure.

Python also provided a convenient mechanism for profiling for performance evaluation. This allows system implementers using the P&U framework to identify and understand bottlenecks and other performance issues.

3. Repository data elements and associated operations can be combined programmatically to identify place-specific opportunistic situations. The following P&U code segment awards a discount coupon to the Pebble owner on an item identified as a specific personal preference if they have visited the store over ten times.

   Tom presents his Pebble to the store Urn as he enters his favorite clothing store. The store Urn presents him with a 15% discount coupon on all shoes in the store that are his size.

```
    if count(Get_Remnant('pid')) >= 10:
      if preference_matchs = PRO_Comp() >= 1:
         item_preference_match = pick(preference_matchs)

/* award discount on matched preference item */
```

As discussed in Section 5.7.5, this research identified the following problematic features that were revealed through the design and implementation of the P&U prototypes.

1. The P&U session characteristic of the Urn waiting for Pebbles using a "block on wait" strategy is inefficient. A more efficient implementation would need to handle the various P&U interface technologies. One approach is the modification of wrapper code to support event driven Pebble and Urn initialization.

2. P&U sessions do not support multiple Pebbles simultaneously. A session feature, such as place adaptation to multiple Pebbles and their owners, is not presently implemented. A framework modification to resolve this issue requires the addition of a Pebble registry sub-system that would track active Pebbles, a Pebble coordination data element in the

Urn Repository, and a common framework function to map and interpret multiple user profiles.

3. While the use of Python is generally supportive of the implementation of Pebbles and Urns, there are caveats.

   Python's sophisticated data structures may make porting of the P&U framework to other languages and development environments difficult. Framework re-design may be required if different development and deployment environments are used. The degree to which P&U framework design was effectively isolated from its implementation using Python is unknown at this time.

4. As described in Section 5.7.1, the prototypes focus on P&U support of the different Pebble classes. At this time, research has not extended Urn capabilities beyond the standalone Urns represented in the scenarios and prototypes. This area of research can be addressed as future work.

Earlier framework designs of the Comm and Transport layers did not provide an isolated physical communication layer. Implementation of new Pebble interfaces required code modifications to higher layers in the P&U architecture. This issue has been resolved with the addition of the Pebble Repository replication step in the evolved P&U design, as presented in Section 4.3.2.2.2.

6.3 The Meaning of it All

A persistent question is "What will be the "killer app" of pervasive computing?" A clear answer is not currently available. But, this section provides some thoughts. As presented in Section 1.1, current pervasive computing research acknowledges there is no clear understanding of how the Internet of Things will make our lives easier and more productive. Existing within the IoT, opportunistic computing is considered one of the next major challenges of pervasive computing.

It is asserted here that continued research in the areas of IoT and OC will lead to the discovery of pervasive computing's "killer app." Based on the discussions of the research questions in Section 6.2, a summary of the contributions of this work follows.

6.3.1 P&U Framework

The implementation and testing of the P&U prototypes described in Section 5.2 validate the P&U framework presented in Section 4.3.2. The P&U framework validates the P&U architectural model discussed in Section 2.2.4. Aspects of the P&U framework contribute to the wider audience of pervasive and ubiquitous computing applications.

As discussed in Section 1.1, researchers acknowledge the need to develop more general infrastructures to facilitate the rapid development and deployment of pervasive computing applications [5, 76]. Although successful IoT research prototypes have been developed, these prototypes are constructed using application- or system-specific "top-to-bottom" software and, often, hardware components [77-79]. It is believe that the need for more general design approaches and infrastructure will be even more important as IoT progresses from research prototypes to deployed production systems.

The following P&U framework characteristics address this need for a more general pervasive computing infrastructure:

- A layered systems architecture with a thin waist"

  The term "thin waist", referenced in Sections 2.2.4, 4.1.1, 5.7.4 and 6.2.2, describes the particular layer of a layered systems architecture that provides commonality connecting the more complex upper application and lower communication layers. The Common Framework layer represents the P&U framework's "thin waist". This *hourglass architecture* is what makes the P&U design elegant and powerful. It allows lower and upper layer technologies to innovate in a largely independent manner without unnecessary constraints. Different pervasive computing applications can be built using the P&U framework without changing the framework's "thin waist."

- Isolated physical layer

  The Interaction Framework layer of the P&U framework defines a standard communication mechanism using interchangeable and extendable wrapper code modules to link Pebbles and Urns. This design feature allows the changing of a P&U system's Pebble class interface by switching only the communication wrapper modules on the Urn. A pervasive computing application built on the P&U framework need only switch wrappers to interact with different devices.

- Design pattern

  As discussed in Section 2.2.4, a design pattern is a software engineering construct that provides a common vocabulary used by software developers to explain a system's design and architecture. Insights gained from the design of the P&U architecture and the implementation and application of the P&U framework, as discussed in Section 6.2.1, define a new design pattern for describing, implementing and deploying Internet of Things and opportunistic computing systems.

6.3.2 Simple IoT Interface

As stated in Section 3.1, the analysis of P&U from a human-computer interface perspective is beyond the scope of this research and dissertation. With this understanding, one contribution of this work is the insight gained from the design and implementation of the P&U framework's tangible interface. The prototype Pebbles and interfaces, as described in Sections 5.3.4 and 5.4.4, are simple and easy to use while providing the necessary functionality for situated interactions in P&U sessions. The design of the P&U Interaction Framework supports interfacing with different Pebble classes. New Pebble class interfaces can be implemented and tested using the wrapper code scheme described in Section 4.3.2.1.

6.3.3 Opportunistic Engine

Sections 4.3.2.3 and 6.2.1.2 discuss the contextual data elements of the P&U Repositories. Common Framework APIs defined in Section 4.3.2.3.2 manage these data elements. As described in Section 5.7.4, these contextual data elements and associated operations can be combined programmatically to identify place-specific opportunistic situations. The work in this area of P&U contributes to better understanding designs of opportunistic engines in OC research.

6.4 Future Research

This section presents recommendations and directions for areas of future research.

- Simultaneous support of multiple, different Pebbles by an Urn would allow it to adjust its place environment based on combined user profiles and preferences.

- In a P&U session, the Urn blocks and waits for Pebbles. More efficient P&U session initialization strategies should be investigated.

- The addition of Urn Repository access to external services would extend P&U's use of cyber foraging strategies. Urn access to cloud resources would extend Urn services.

- Common Precept Name is an example of a standard naming convention linking Pebble Precept requests to Urn services. Additional formal data specifications and metadata for Repository objects like user profile and place preference data would result in more powerful matching capabilities.

- A hybrid Pebble is capable of performing more sophisticated actions while still maintaining a minimal resource configuration. Multiple interfaces on a single Pebble form factor can associate separate external services by combining separate pieces of information. P&U support of hybrid Pebbles should be investigated.

- Certain P&U interactions involve the use and transfer of identifying information describing the Pebble owner. Privacy and security features of P&U should be explored and implemented to ensure Pebble owner confidentiality.

References

[1] C. R. Schoenberger. (2002, March 18) The Internet of Things. *Forbes Magazine*.

[2] N. Gershenfeld, *When Things Start to Think*: Henry Holt and Company, 1999.

[3] D. Uckelmann, M. Harrison, and F. Michahelles, *Architecting the Internet of Things*: Springer, 2011.

[4] P. Lucas, "The Trillion-Node Network," MTR-00001, 1999.

[5] F. Mattern and C. Floerkemeier, "From the Internet of Computers to the Internet of Things," in *From Active Data Management to Event-Based Systems and More*, S. Kai, P. Ilia, and G. Pablo, Eds., ed: Springer-Verlag, 2010, pp. 242-259.

[6] M. Conti and M. Kumar, "Opportunities in Opportunistic Computing," *Computer,* vol. 43, pp. 42-50, 2010.

[7] R. Balan, J. Flinn, M. Satyanarayanan, S. Sinnamohideen, and H.-I. Yang, "The Case for Cyber Foraging," presented at the Proceedings of the 10th workshop on ACM SIGOPS European Workshop, Saint-Emilion, France, 2002.

[8] "relevant," in *in Dictionary.com Unabridged*, http://dictionary.reference.com/browse/relevant. ed: Source Location: Random House, Inc. Available: http://dictionary.reference.com. Accessed: March 04, 2012.

[9] D. Bohus and E. Horvitz, "Dialog in the Open World: Platform and Applications," presented at the Proceedings of the 2009 International Conference on Multimodal Interfaces, Cambridge, Massachusetts, USA, 2009.

[10] "omphalos," in *in Dictionary.com Unabridged*, http://dictionary.reference.com/browse/omphalos. ed: Source Location: Random House, Inc. Available: http://dictionary.reference.com. Accessed: February 16, 2012.

[11] M. Weiser, R. Gold, and J. S. Brown, "The Origins of Ubiquitous Computing Research at PARC in the Late 1980s," *IBM Systems Journal,* vol. 38, pp. 693-696, 1999.

[12] E. Larson, "Introduction to GumStix Computers," University of West Florida2006.

[13] E. Rukzio, G. Broll, and S. Wetzstein, "The Physical Mobile Interaction Framework (PMIF)," Technical Report LMU-MI-2008-2, 2008.

[14] V. Savikko, "Design Patterns in Python," in *Proceedings of the 6th International Python Conference, CNRI*, 1997, pp. 63-68.

[15] Z. Cheng and D. Budgen, "What Do We Know about the Effectiveness of Software Design Patterns?," *Software Engineering, IEEE Transactions on,* vol. 38, pp. 1213-1231, 2012.

[16] E. Agerbo and A. Cornils, "How to Preserve the Benefits of Design Patterns," presented at the Proceedings of the 13th ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, Vancouver, British Columbia, Canada, 1998.

[17] S. Akhshabi and C. Dovrolis, "The Evolution of Layered Protocol Stacks Leads to an Hourglass-Shaped Architecture," *SIGCOMM Comput. Commun. Rev.,* vol. 41, pp. 206-217, 2011.

[18] E. Rukzio, G. Broll, K. Leichtenstern, and A. Schmidt, "Mobile Interaction with the Real World: an Evaluation and Comparison of Physical Mobile Interaction Techniques," presented at the Proceedings of the 2007 European Conference on Ambient Intelligence, Darmstadt, Germany, 2007.

[19] R. Kazhamiakin, V. Kerhet, M. Paolucci, M. Pistore, and M. Wagner, "Discovery Pervasive Services based on their Expected Use," presented at the 3rd International SMR2 2009 Workshop on Service Matchmaking and Resource Retrieval in the Semantic Web, Washington, DC, 2009.

[20] Y. Yamato, H. Ohnishi, and H. Sunaga, "Study of Service Processing Agent for Context-Aware Service Coordination," in *IEEE International Conference on Services Computing, SCC '08*, 2008, pp. 275-282.

[21] H. Ishii and B. Ullmer, "Tangible Bits: Towards Seamless Interfaces Between People, Bits and Atoms," presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Atlanta, Georgia, United States, 1997.

[22] O. Shaer, N. Leland, E. H. Calvillo-Gamez, and R. J. K. Jacob, "The TAC Paradigm: Specifying Tangible User Interfaces," *Personal Ubiquitous Comput.,* vol. 8, pp. 359-369, 2004.

[23] G. Abowd, A. Dey, P. Brown, N. Davies, M. Smith, and P. Steggles, "Towards a Better Understanding of Context and Context-Awareness," in *Handheld and Ubiquitous Computing*. vol. 1707, H.-W. Gellersen, Ed., ed: Springer Berlin / Heidelberg, 1999, pp. 304-307.

[24] M. McCullough, "On Typologies of Situated Interaction," *Hum.-Comput. Interact.,* vol. 16, pp. 337-349, 2001.

[25] A. Schmidt, W. V. d. Velde, and G. Kortuem, "Situated Interaction in Ubiquitous Computing," presented at the CHI '00 Extended Abstracts on Human Factors in Computing Systems, The Hague, The Netherlands, 2000.

[26] V. Waller and R. B. Johnston, "Making Ubiquitous Computing Available," *Commun. ACM,* vol. 52, pp. 127-130, October 2009.

[27] G. Broll, S. Siorpaes, E. Rukzio, M. Paolucci, J. Hamard, M. Wagner, and A. Schmidt, "Supporting Mobile Service Usage through Physical Mobile Interaction," presented at the Pervasive Computing and Communications, 2007. PerCom '07. Fifth Annual IEEE International Conference on, 2007.

[28] P. Saint-Andre, K. Smith, and R. TronCon, *XMPP: The Definitive Guide : Building Real-Time Applications with Jabber Technologies*: O'Reilly, 2009.

[29] J. Messeter, "Place-Specific Computing: A Place-centric Perspective for Digital Designs," *International Journal of Design,* vol. 3, pp. 29-41, April 2009.

[30] D. Seamon, "A Lived Hermetic of People and Place: Phenomenology and Space Syntax," presented at the 6th International Space Syntax Symposium, Istanbul, Turkey, 2007.

[31] P. M. Todd and G. Gigerenzer, "Environments That Make Us Smart: Ecological Rationality," *Current Directions in Psychological Science,* vol. 16, pp. 167-171, 2007.

[32] V. Kalnikaite, Y. Rogers, J. Bird, N. Villar, K. Bachour, S. Payne, P. M. Todd, J. Schning, A. Krger, and S. Kreitmayer, "How to Nudge in Situ: Designing Lambent Devices to Deliver Salient Information in Supermarkets," presented at the Proceedings of the 13th International Conference on Ubiquitous computing, Beijing, China, 2011.

[33] M. Conti, S. Giordano, M. May, and A. Passarella, "From Opportunistic Networks to Opportunistic Computing," *IEEE Communications Magazine,* vol. 48, pp. 126-139, Sept 2010.

[34] J. Peek. (2008, October 15) Filenames by Design, Part One. *Linux Magazine*. Available: http://www.linux-mag.com/id/7158/

[35] M. Mamei, R. Quaglieri, and F. Zambonelli, "Making Tuple Spaces Physical with RFID Tags," presented at the Proceedings of the 2006 ACM Symposium on Applied Computing, Dijon, France, 2006.

[36] S. Kolahdooz, S. Rahmani, and M. Sharifi, "Discovering Resources in Tuple-Based Pervasive Systems using Resource-Aware Routing," presented at the IEEE International Performance, Computing and Communications Conference, Austin, Texas, 2008.

[37] D. Balzarotti, P. Costa, and G. P. Picco, "The LighTS Tuple Space Framework and its Customization for Context-Aware Applications," *Web Intelli. and Agent Sys.,* vol. 5, pp. 215-231, April 2007.

[38] A. Nori, "Mobile and Embedded Databases," presented at the Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, Beijing, China, 2007.

[39] P. Bonnet, J. Gehrke, and P. Seshadri, "Towards Sensor Database Systems," presented at the Proceedings of the Second International Conference on Mobile Data Management, 2001.

[40] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The Many Faces of Publish/Subscribe," *ACM Comput. Surv.,* vol. 35, pp. 114-131, 2003.

[41] D. Locke, "MQ Telemetry Transport (MQTT) V3.1 Protocol Specification," IBM Corporation, developWorksAugust 2010.

[42] *MQ Telemetry Transport*. Available: http://mqtt.org/

[43] U. Hunkeler, T. Hong Linh, and A. Stanford-Clark, "MQTT-S - A Publish/Subscribe Protocol for Wireless Sensor Networks," presented at the 3rd International Conference on Communication Systems Software and Middleware and Workshops, 2008. COMSWARE 2008., 2008.

[44] J. Krumm, N. Davies, and C. Narayanaswami, "User-Generated Content," *IEEE Pervasive Computing,* vol. 7, pp. 10-11, 2008.

[45] F. Girardin, F. Calabrese, F. D. Fiore, C. Ratti, and J. Blat, "Digital Footprinting: Uncovering Tourists with User-Generated Content," *IEEE Pervasive Computing,* vol. 7, pp. 36-43, October 2008.

[46] M. Bernstein, M. V. Kleek, D. Karger, and M. C. Schraefel, "Information Scraps: How and Why Information Eludes Our Personal Information Management Tools," *ACM Trans. Inf. Syst.,* vol. 26, pp. 1-46, Sept 2008.

[47] M. Mamei and F. Zambonelli, "Pervasive Pheromone-Based Interaction with RFID Tags," *ACM Trans. Auton. Adapt. Syst.,* vol. 2, p. 4, 2007.

[48] F. Zhu, M. W. Mutka, and L. M. Ni, "Service Discovery in Pervasive Computing Environments," *IEEE Pervasive Computing,* vol. 4, pp. 81-90, Oct-Dec 2005.

[49] F. Zhu, W. Zhu, M. W. Mutka, and L. M. Ni, "Service Discovery Architecture and Protocol Design for Pervasive Computing," in *Advanced Design Approachers to Emerging Software Systems: Principles, Methodologies and Tools*, X. Liu and Y. Li, Eds., ed: IGI Global, 2011, pp. 83-101.

[50] K. Rasch, F. Li, S. Sehic, R. Ayani, and S. Dustdar, "Context-Driven Personalized Service Discovery in Pervasive Environments," *World Wide Web,* vol. 14, pp. 295-319, July 2011.

[51] C. Doulkeridis, N. Loutas, and M. Vazirgiannis, "A System Architecture for Context-Aware Service Discovery," *Electron. Notes Theor. Comput. Sci.,* vol. 146, pp. 101-116, January 2006.

[52] P. Patel and S. Chaudhary, "Context Aware Semantic Service Discovery," presented at the Proceedings of the 2009 World Conference on Services - II, 2009.

[53] M. Weber, T. Roth-Berghofer, V. Hudlet, H. Maus, and A. Dengel, "Context-Aware Service Discovery Using Case-Based Reasoning Methods," presented at the Proceedings of the 32nd Annual German Conference on Advances in Artificial Intelligence, Paderborn, Germany, 2009.

[54] L. Yu, S. Luo, and A. Glenstrup, "Rough Sets Based Context-Aware Service Discovery Framework," presented at the Proceedings of the 2010 International Conference on Service Sciences, 2010.

[55] L. Youseff, M. Butrico, and D. Da Silva, "Toward a Unified Ontology of Cloud Computing," in *Grid Computing Environments Workshop, 2008. GCE '08*, 2008, pp. 1-10.

[56] M. Hofer and G. Howanitz, "The Client Side of Cloud Computing," ed: University of Strausburg, 2009.

[57] M. D. Kristensen, M. B. Kjaergaard, T. Toftkjaer, S. Bhattacharya, and P. Nurmi, "Improving Pervasive Positioning Through Three-Tier Cyber Foraging," presented at the 2011 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011.

[58] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The Case for VM-Based Cloudlets in Mobile Computing," *IEEE Pervasive Computing,* vol. 8, pp. 14-23, 2009.

[59] J. Flinn, P. SoYoung, and M. Satyanarayanan, "Balancing Performance, Energy, and Quality in Pervasive Computing," presented at the 22nd International Conference on Distributed Computing Systems, 2002. Proceedings., 2002.

[60] T. Abdelzaher, C. Qing, R. Ganti, D. Henriksson, M. Khan, H. Jin, H. Chengdu, P. Jayachandran, L. Hieu Khae, L. Liqian, and T. Yu-En, "Towards a Layered Architecture for Object-Based Execution in Wide-Area Deeply Embedded Computing," in *Object and Component-Oriented Real-Time Distributed Computing, 2007. ISORC '07. 10th IEEE International Symposium on*, 2007, pp. 133-140.

[61] J. Akehurst, I. Koprinska, K. Yacef, L. Pizzato, J. Kay, and T. Rej, "Explicit and Implicit User Preferences in Online Dating," presented at the Proceedings of the 15th International Conference on New Frontiers in Applied Data Mining, Shenzhen, China, 2012.

[62] J. Postel, "Transmission Control Protocol," 1981.

[63] S. Spiekermann, *User Control in Ubiquitous Computing: Design Alternatives and User Acceptance*: Shaker, 2007.

[64] (November 19, 2012). *Python 2.5.2 Release*. Available: http://http://www.python.org/download/releases/2.5.2/

[65] (November 19, 2012). *lubuntu*. Available: http://http://lubuntu.net/

[66] (November 19, 2012). *wxPython*. Available: http://http://www.wxpython.org/

[67] (November 19, 2012). *ZBar Bar Code Reader*. Available: http://http://zbar.sourceforge.net/

[68] A. Haerter. (2010). *QR Code Generator goQR.me*. Available: http://goqr.me/

[69] H. Kato and K. T. Tan, "Pervasive 2D Barcodes for Camera Phone Applications," *IEEE Pervasive Computing,* vol. 6, pp. 76-85, 2007.

[70] (November 19, 2012). *The Python Profilers*. Available: http://http://docs.python.org/2/library/profile.html

[71] (November 19, 2012). *ubuntu manuals - udev*. Available: http://manpages.ubuntu.com/manpages/oneiric/man7/udev.7.html

[72] (November 19, 2012). *PyBluez*. Available: http://code.google.com/p/pybluez/

[73] S. B. Handurukande, S. Ganguly, and S. Bhatnagar, "Fast Bluetooth Service Discovery for Mobile Peer-to-Peer Applications," ed, 2006.

[74] W. Ryan, J. Derek, C. Trevor, and K. D. Charles, "Rapid Heterogeneous Connection Establishment: Accelerating Bluetooth Inquiry Using IrDA," in *Proceedings of the Third Annual IEEE Wireless Communications and Networking Conference (WCNC'02)*, 2002.

[75] (November 19, 2012). *python*. Available: http://www.python.org/

[76] P. Wozniak and A. Romanowski, "Everyday Problems vs. UbiComp: A Case Study," presented at the Proceedings of the 2nd International Conference on Web Intelligence, Mining and Semantics, Craiova, Romania, 2012.

[77] R. Caceres and A. Friday, "Ubicomp Systems at 20: Progress, Opportunities, and Challenges," *Pervasive Computing, IEEE,* vol. 11, pp. 14-21, 2012.

[78] G. D. Abowd, "What Next, Ubicomp? Celebrating an Intellectual Disappearing Act," 2012.

[79] B. S. Chouhan and C. K. Chhatlani, "A Study of Ubiquitous Computing Applications & Its Framework," *International Journal of Computer Applications & Natural Sciences,* p. 53.

Appendix

Appendix A: Initial P&U Design Elements

This appendix presents more detailed specification of portions of the initial P&U design presented in Section 4.2.2.

A.1 Data Structures

PM_Structure		
Description: Data structure containing parsed Pebble Message elements		
Data Elements	Str	PID
	Str	CID
	Str	UID
	Str	Svc_Req
	Int	Msg_Hash

A.2 P&U Functions

URA_Pebble(PM_Structure)			
Description: Requests Pebble via UPCA API call–passes PM_Structure			
Parameters	Output	Str	PM_Structure
Return Value: Null			

UR_Process_Pebble(Pebble_Message)			
Description: Deconstructs Pebble_Message, loads PM_Structure, log file			
Parameters	Input	Str	Pebble_Message
	Output	Str	PM_Struct
Return Value: Null			

UPCA_Get_Pebble(Pebble_Message)			
Description: Gets Pebble_Message via wrapper code, performs hash check			
Parameters	Output	Str	Pebble_Message
Return Value: Null			

UCLA_Wrapper(Pebble_Message)			
Description: Invokes one of the physical communication link routines			
Parameters	Output	Str	Pebble_Message
Return Value: Null			

A.3 P&U Tables

Pebble of Interest

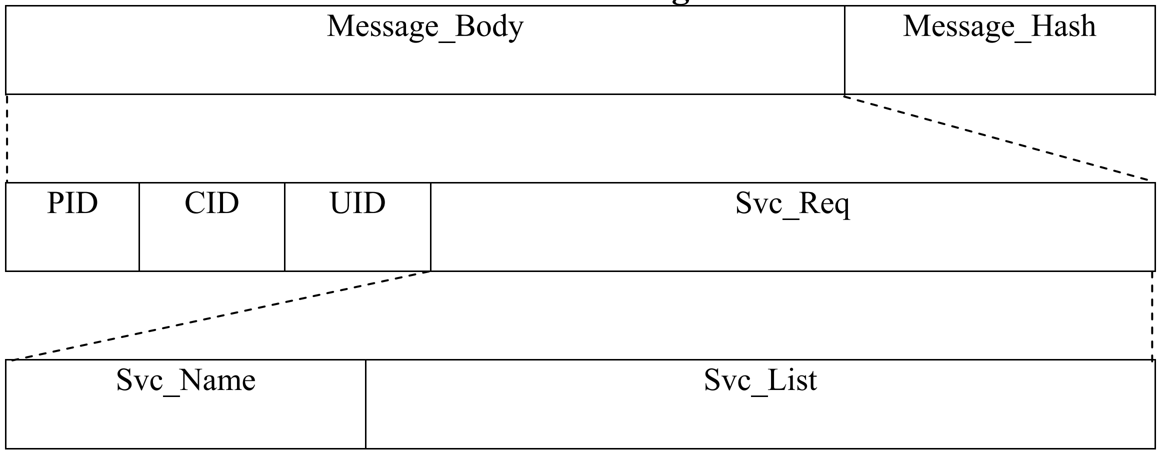
Name: POI	Location: Urn Repository
Column Name	Type
PID	Str

Pebble Log

Name: Plog	Location: Urn Repository
Column Name	Type
PID	Str
CID	Str
UID	Str
Svc_Req	Str
Msg_Hash	Int
TimeDate	Str

A.4 P&U Message

Pebble Message



Pebble Message			
Position	Length	Name	Notes
0-71	72	Message_Body	Body of Pebble Message
72-79	8	Message_Hash	Hash of Message_Body

Message Body			
Position	Length	Name	Notes
0-3	4	PID	Pebble Identifier
4-7	4	CID	Pebble Class Identifier
8-11	4	UID	Pebble Owner Identifier
12-71	60	Svc_Req	Service Request to Urn

Svc Req			
Position	Length	Name	Notes
12-19	8	Svc_Name	Requested Service Name
20-71	52	Svc_List	Service Request Data

Appendix B: P&U Framework Layers

B.1 Interaction Framework

WRAP_Comm(urn_interface)			
Description: Provides Pebble Repository access to the Urn Common Framework by means of wrapper functions			
Parameters	Input	Str	urn_interface
			"QR"
			"USB"
			"BT"
Return Value: object returned from called wrapper code			

Get_QR()			
Description: Transfers text from Read-Only QR Pebble			
Parameters			
Return Value: decoded QR text string			

Link_USB()			
Description: Links the Read/Write Pebble Repository to the Urn Repository			
Parameters			
Return Value: return code from filesystem mount operation (0=successful, -1=unsuccessful)			

Transfer_BT()			
Description: Transfers mirrored Smart Pebble Repository to and from the Urn Repository			
Parameters			
Return Value: file transfer status (0=successful, -1=unsuccessful)			

B.2 Common Framework

B.2.1 Transport Functions

MIRROR_Load_PR(urn_interface)			
Description: Loads a mirrored Pebble Repository into the Urn Repository at the beginning of the P&U session			
Parameters	Input	str	urn_interface
			"QR"
			"USB"
			"BT"
Return Value: Null			

MIRROR_Store_PR(urn_interface)			
Description: Sends mirrored image back to Pebble at the end of the P&U session			
Parameters	Input	str	urn_interface
			"QR"
			"USB"
			"BT"
Return Value: Null			

MIRROR_Send_PR()			
Description: Sends mirrored Pebble Repository to the Urn at the beginning of the P&U session			
Parameters			
Return Value: Null			

MIRROR_Receive_PR()			
Description: Stores mirrored image back onto the Pebble Repository at the end of the P&U session			
Parameters			
Return Value: file transfer status (0=successful, -1=unsuccessful)			

B.3 Application Framework

B.3.1 Common Framework APIs

PRE_Precept()			
Description: Executes all matching Precepts			
Parameters			
Return Value: Null			

PRO_Comp()			
Description: Performs comparison of user profiles and place preferences			
Parameters			
Return Value: Null			

PRO_Update()			
Description: Updates user profile in Pebble Repository			
Parameters			
Return Value: Null			

REM_Get_Remnant(target)			
Description: Retrieves Urn and Pebble remnants			
Parameters	Input	Str	target
			“pebble_most_recent”
			“pebble_list”
			“urn_most_recent”
			“urn_list”
Return Value: “most recent” request returns Remnant in dictionary structure “list” request returns list of Remnant identifiers			

PUU_UIM(item)			
Description: Retrieves item from Urn UIM structure			
Parameters	Input	Str	item
			“uid”
Return Value: requested item as str data type			

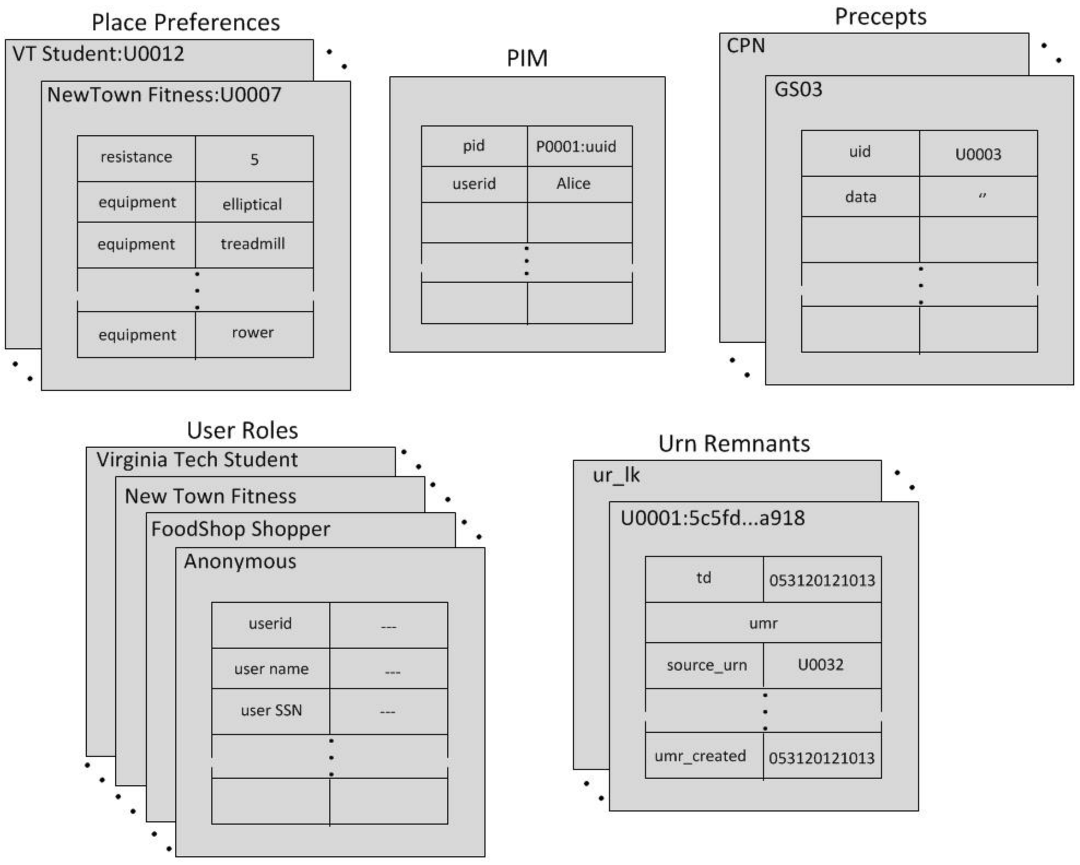
PUU_PIM(item)			
Description: Retrieves item from Pebble PIM structure			
Parameters			
			“pid”
			“cid”
			“userid”
Return Value: requested item as str data type			

SP_Select_Role()			
Description: Performs role selection function on Smart Pebble			
Parameters			
Return Value: Null			

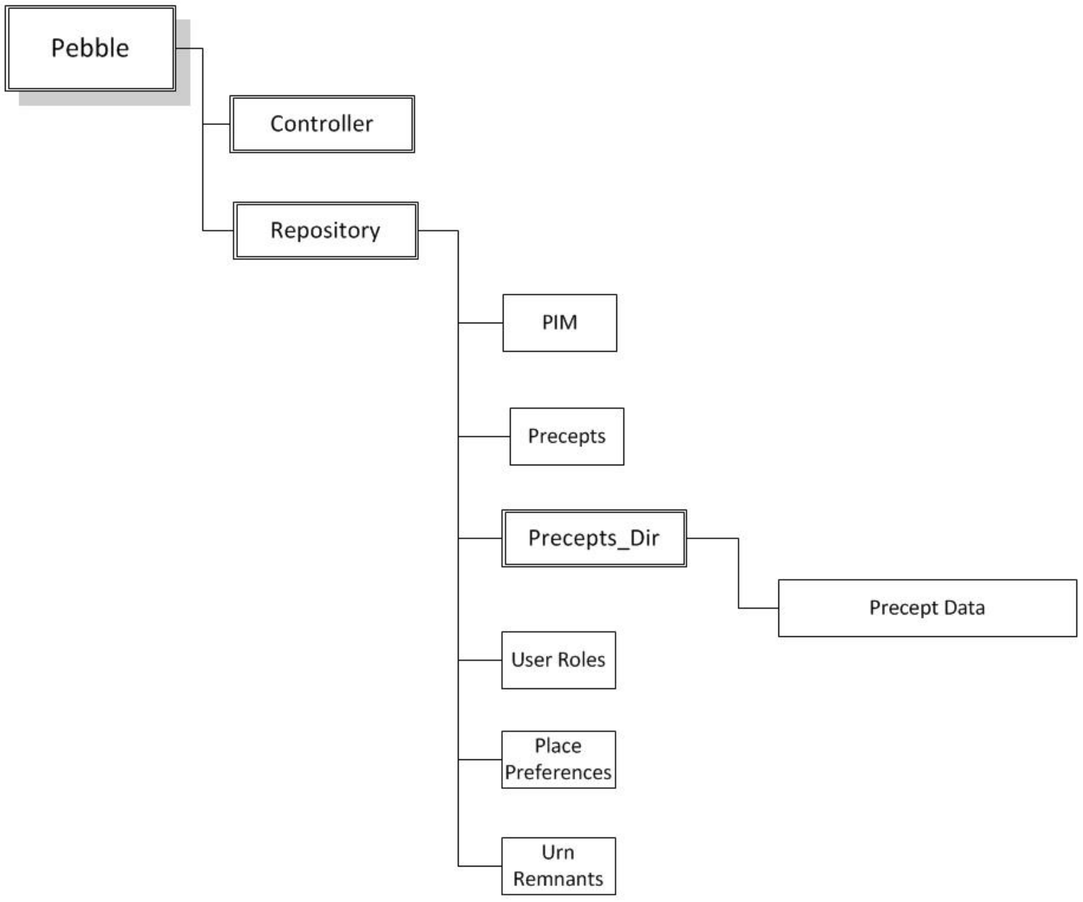
UI_Text_Out(text)			
Description: Displays text on Smart Pebble or Urn display			
Parameters	Input	Str	text
Return Value: Null			

Appendix C: P&U Repository Data Dictionary

C.1 Pebble Repository



C.2 Pebble Directory Structure



C.3 Pebble Repository Data Elements

PIM		
Pebble Initialization Message		
Item	**Type**	**Description**
pid	str	Pebble id
userid	str	Pebble owner id

PIM – Pebble Initialization Message

Data type: Python Dictionary

Format
{'pid': pid, ‘userid’:Pebble owner id}

Keys:
pid – unique Pebble id
userid – Pebble owner id

Example:
{'pid': 'P0005:d5e46451-b6bc-4407-b727-2b9f61b68851', 'userid': 'bill'}

Pebble Precepts

Pebble service requests

Item	Type	Description
CPN	str	Common Precept Name
uid	str	Urn id
data	str	Pebble data for Urn Precept

Precepts – Service Requests

Data type: Python Dictionary of Dictionaries

Format
{CPN: {'uid': Urn id, 'data’:data}

Keys:
CPN – Common Precept Name
uid – Urn id
data – Pebble data for Urn Precept (ex. grocery store list)

Example:
{'GS03': {'uid': 'U0003', ‘data’:’’}}

Urn Remnants		
A representation of an Urn’s interaction with a particular Pebble		
Item	**Type**	**Description**
ur_lk	str	Instance name linking to Urn remnant equivalent
td	str	Remnant creation time/date stamp
umr	str	Userid of Pebble owner
source_urn	str	Urn generating the UMR
target_urn	str	Recipient Urn of the UMR
umr_message	str	UMR message
umr_created	Str	UMR creation time/date stamp

Urn Remnants stored in Pebble Repository

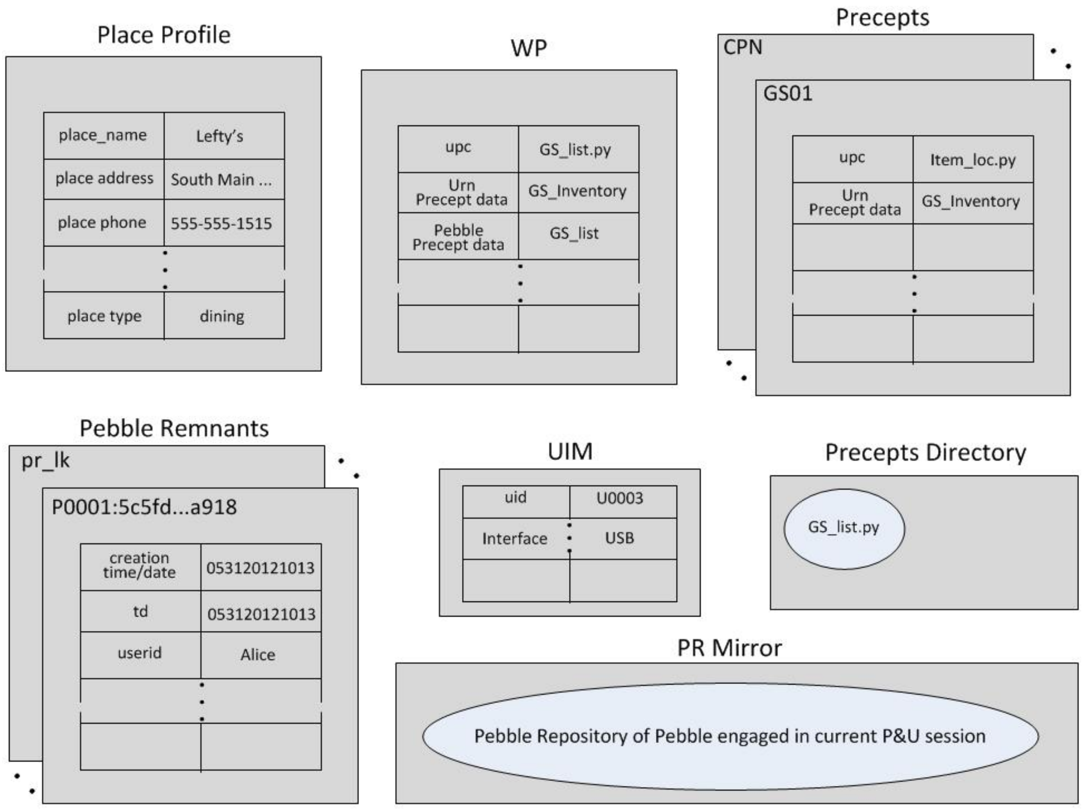
Data type: Python Dictionary of Dictionaries

Format
{ur_lk: {'td': current time/date stamp, ‘umr’:{‘target_urn’:target Urn,’source_urn’:source Urn,’urn_message’:Urn message,’message_created’:time/date stamp}}}

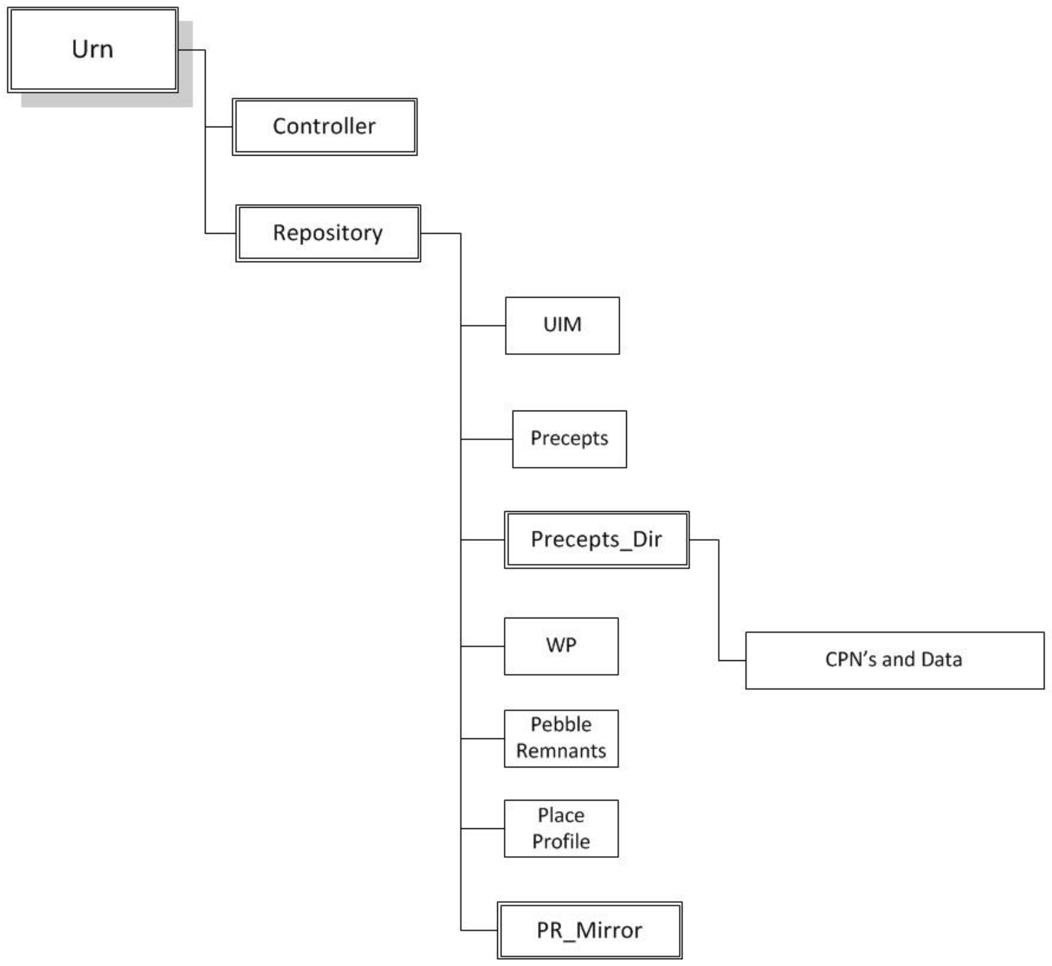
Keys:
ur_lk – unique Urn Remnant id
td – current time/date stamp
umr – UMR structure
target_urn – UMR destination Urn
source_urn – UMR source Urn
urn_message – UMR message
message_created – time/date/stamp when UMR created

Example:
{'U0002:d520a97c-0c98-4c86-a497-8714430d0ccd': {'td': '20120925213909', 'umr': {'target_urn': 'U0043', 'source_urn': 'U0032','urn_message': 'remember user', 'message_created': '20120925213909'}}}

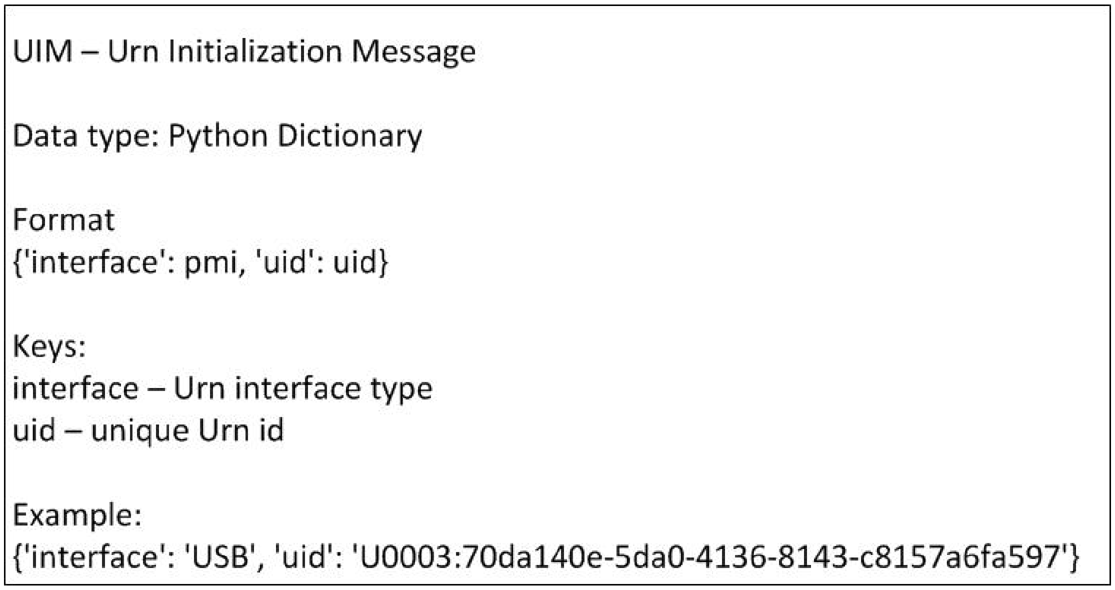
C.4 Urn Repository



C.5 Urn Directory Structure



C.6 Urn Repository Data Elements


UIM – Urn Initialization Message

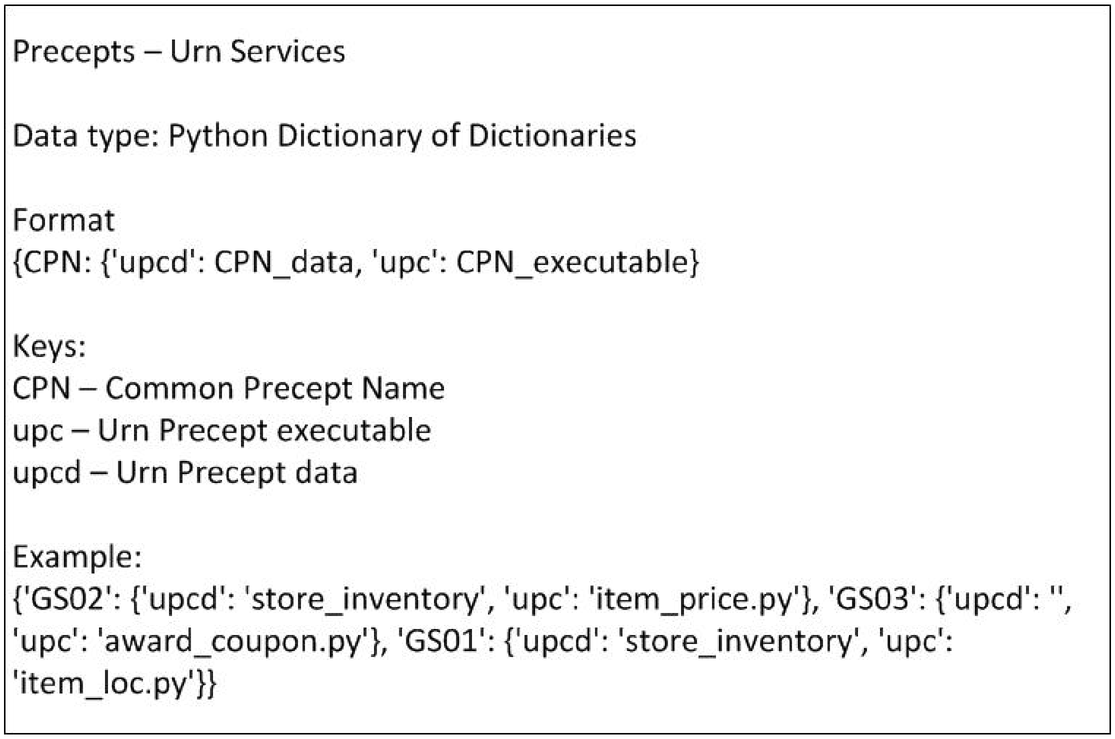
Data type: Python Dictionary

Format
```
{'interface': pmi, 'uid': uid}
```

Keys:
interface – Urn interface type
uid – unique Urn id

Example:
```
{'interface': 'USB', 'uid': 'U0003:70da140e-5da0-4136-8143-c8157a6fa597'}
```

UIM		
Urn Initialization Message		
Item	**Type**	**Description**
interface	str	Instance name linking to Pebble remnant equivalent
uid	str	Remnant creation time/date stamp



Precepts – Urn Services

Data type: Python Dictionary of Dictionaries

Format
{CPN: {'upcd': CPN_data, 'upc': CPN_executable}

Keys:
CPN – Common Precept Name
upc – Urn Precept executable
upcd – Urn Precept data

Example:
{'GS02': {'upcd': 'store_inventory', 'upc': 'item_price.py'}, 'GS03': {'upcd': '', 'upc': 'award_coupon.py'}, 'GS01': {'upcd': 'store_inventory', 'upc': 'item_loc.py'}}

Urn Precepts		
Urn services		
Item	**Type**	**Description**
CPN	str	Common Precept Name
upc	str	Urn Precept executable
upcd	str	Urn Precept data

Pebble Remnants store in Urn Repository

Data type: Python Dictionary of Dictionaries

Format
{pr_lk: {'td': current time/date stamp, 'userid':Pebble owner’s id}

Keys:
pr_lk – unique Pebble Remnant id
td – current time/date stamp
userid – Pebble owner’s id

Example:
{'P0005:d3ddbe98-51db-4b7e-97a4-f92eef9cfbd4:3d1293d2-ffbb-4a26-ac8f-47d8eac8a147': {'td': '20121123062321', 'userid': 'tswift'}}

Pebble Remnants		
A representation of a Pebble’s interaction with a particular Urn.		
Item	**Type**	**Description**
pr_lk	str	Instance name linking to Pebble remnant equivalent
td	str	Remnant creation time/date stamp
userid	str	Pebble owner’s id

Appendix D: P&U Application Developer Notes

D.1 Developer Tools

The P&U application developer uses these tools to create and maintain Pebbles and Urns.

Function	Description
Create_UIM()	Create Urn initialization message
Print_UIM()	Print Urn initialization message
Create_PIM()	Create Pebble initialization message
Print_PIM()	Print Pebble initialization message
Create_Urn_Precept()	Create Urn precept
Print_Urn_Precept()	Print Urn precept
Create_Pebble_Precept()	Create Pebble precept
Print_Pebble_Precept()	Print Pebble precept
Create_Place_Profile()	Create place profile
Print_Place_Profile()	Print place profile
Create_User_Profile()	Create user profile
Print_User_Profile()	Print user profile
Print_Pebble_Remnant()	Print Pebble remnant
Print_Urn_Remnant()	Print Urn remnant
Change_Urn_Interface()	Changes the Urn's physical interface

D.2 How to Create a Pebble

The Read-Only Pebble used in the prototypes discussed in Chapter 5 is based on a QR code. Text following the Pebble Repository format in Section 5.3.2 is encoded into a QR code using the procedure discussed in Section 5.1.1.

A Read/Write Pebble is primarily a Pebble Repository. The repository is created in the following steps. This example assumes a USB Pebble mounted on a system running the Unix operating system.

1. Create directory /Pebble/Repository
2. Make the Pebble's PIM with the Create_PIM command

Create_PIM ‘P0005’ ‘tswift’ ‘USB’

3. Make the Precepts file. Repeat for multiple CPN’s.

   Create_Pebble_Precept ‘GS03’ ‘U0003’

4. Create user profile.

   Create_User_Profile ‘tswift’ ‘Tom Swift’ ‘222-33-4444’

The developer tools run directly on the Smart Pebble. Follow the same steps to create the Smart Pebble Repository.

D.3 How to Create an Urn

The Urn Repository is created following the same steps as those for Pebble Repository creation described in Appendix D.2. Urn applications are developed using the API’s presented in Table 16 and Appendix B.3.1. This example Urn application performs a check for an Urn message, processes Precepts, updates Remnants on the Pebble and the Urn. At completion of session, the Urn waits for another Pebble.

```
# Wait for Pebble
   while True:

# Mirror Pebble Repository onto Urn Repository
      comm.MIRROR.Load_PR()

# Process Urn Message Relay
      common.UMR.UMR_Message()

# Execute Precept Matches
      common.PRE.PRE_Precept('auto’)

# Update Pebble and Urn Remnants
      common.REM.REM_Put_Remnant(‘pebble’)
      common.REM.REM_Put_Remnant(‘urn’)

# Store Pebble Repository mirror back onto Pebble
      comm.MIRROR.Store_PR()
```

Appendix E: Glossary

Architecture – fundamental organization of a system embodied in its components, their relationship to each other and the environment.

Common Precept Name (CPN) – a standard naming convention that associates a Pebble Precept request with an Urn Precept service.

Component – an architectural entity that encapsulates a subset of the systems functionality and/or data, restricts access to that subset via an explicitly defined interface and has explicitly defined dependencies on its required execution context.

Connector – an architectural element tasked with effecting and regulating interactions among components.

Configuration – a set of specific associations between the components and connectors of a software system's architecture.

Framework – a bridge between an architectural style and a set of implementation techniques.

Opportunistic engine – a concept component of opportunistic computing that is capable of identifying opportunistic situations. P&U demonstrates a rudimentary version of this concept with Precept processing and Remnant traversal.

Orchestration – the coordination and deployment of services considering the contextual framework within which a service request is made.

Pattern – a named collection of architectural design decisions that are applicable to a recurring design problem, parameterized to account for different software development contexts in which that problem appears.

Publish/subscribe – a messaging system in which messages are exchanged without using targeted recipients. Messages containing references of specific interests are published and subscribers obtain messages by expressing an interest.

Service Composition – the process of constructing a composite service from atomic services in order to achieve a specific task.

SOA – Service-Oriented Architecture – software engineering design principles emphasizing the development of software as interoperable services.

Style – a named collection of architectural design decisions that are applicable in a given development context, constrain architectural design decisions that are specific to a particular system within that context and elicit beneficial qualities in each resulting system.

Thin-waist – The Internet architecture is a layered architecture with specific applications at the top and physical layer protocols at the bottom. The middle layer (network layer) contains only one protocol (IP) thus representing a "thin-waist" architecture. IP in the "thin waist" of the Internet protocol provides the commonality connecting the more complex upper and lower layers.

Working Precept (WP) – working storage on the Urn where the executable command string is built during Precept processing.