Abstract—The architecture of a software system is concerned with the top-level decomposition of the system into its main components. We have accessed the issues about the key point is how to link some requirements to some architecture decisions such as creation of components and connectors with special features, adoption of architectural styles and patterns upon which the design will be built up, and establishment of communication protocols through which the whole system can be integrated. We provide software architecture centered design process called SADPBA. (Requirements, architecture and system details). We present the MEECS (Mobile Embedded E-Commerce System) by which we perform the explorative research on applying agent into mobile e-commerce.

Index Terms—Software architecture, reuse, architectural design space and rules, software architecture centered design process, mobile embedded E-commerce system.

I. INTRODUCTION

Software architecture is the study of the large-scale structure and performance of software systems [Shaw 89]. Important aspects of a system’s architecture include the division of functions among system modules, the means of communication between modules, and the representation of shared information. The architectural alternatives available to a system designer can be described and classified by constructing a design space. Within a design space, we can formulate design rules that indicate good and bad combinations of choices. Such rules can be used to select an appropriate system design based on functional requirements. A valuable contribution will be made if the rules can help a journeyman designer to make choices comparable to those that a master designer would make or even just help the journeyman to choose a reasonable design with no major errors. With sufficient experience, a set of such rules may become complete and reliable enough to serve as the basis for automated system design, but the rules can be of practical use long before that stage is reached. In this paper section 2 provides Software architecture centered design process is called SADPBA, overview of SADPBA, section 3 presents Trace mechanism in SADPBA. Section 4 presents Case study of MEECS and also provide application of SADPBA on MEECS.

II. SADPBA

In this section, we provide our software architecture centered design process; called SADPBA (Software Analysis and Design Process Based on Architecture). This process uses the ideas of design space in the mobile-based collaboration platform. In the following, we give the introduction to its overview, design space application and track relationships in them. This process follows this pattern, which is the topic of the next section. The overview of SADPBA is shown in Fig.1. Intuitively, through the analysis of SH, Res, Act and Cons of actions next to each other, designers can judge the correctness of transfer between design spaces.

And define the map relationships as: DS1 and DS2 are two design spaces (they may be the same one), if a rule exists, expressed as f, which makes any element $\alpha \in DS_1$ has a counterpart $\beta$ in DS2, we say that f is a map from DS1 to DS2. This relationship is expressed as $f: DS_1 \rightarrow DS_2$ and $f(\alpha) = \beta$.

In this manner, it is possible to implement a "design machine" that is fed with requirements and give design results. This machine can also choose the best one, if we predetermine an evaluation formula and assign weight to different dimensions. To achieve this point, we need gather the rules for mapping, normally in the way of comparing and summarizing the designs under design space specifications. Therefore, we own the capability of automatically judging whether a design is reasonable and potential pitfalls incurred.
by the hidden shortcomings.

III. TRACE MECHANISM IN SADPBA

For a tool capable of automatic design, it is important to judge whether its output is good or not. A validation achieves this, which calls for the information about mapping between every design spaces. This is why we need to extract the "trace" relationship in SADPBA.

Trace means a bidirectional relationship between two elements in one or multiple design spaces, which defined by a certain rules. More formally, in design spaces, if element α can be tracked to the element β, we say that element α and β have traceable relationship, expressed as a Trace to α β. (Since this relationship is bidirectional, given a Trace to α β, Trace to a holds definitely.) In SADPBA, we categorize three kinds of traceable relationships. They come from the completeness of design spaces used in SADPBA. The first one stands between DSF and DSA: Given DSF is a Function Design Space and DSA is an Architecture Design Space, and f is a rule of map between them. Only when any element α ∈ DSF and all its dependencies can be mapped to the element in DSA, DSv is complete to DSA. Another one exists between DSA and DSs: Given DSA is an Architecture Design Space and DSs is an System Design Space, and f is a rule of map between them. Only when any element α ∈ DSA and all its dependencies can be mapped to the element in DSs, DSA is complete to DSs. Purely from mathematics, completeness means every element in a domain of a map can find a counterpart in its range. In design space, completeness guarantees that given an input in one design space we can find a result in the next design space for sure. The whole design process of SADPBA thus behaves determinably, which is the foundation of automatic design capable tool. SADPBA employs the sequence-based specification process. Each sequence indicates a use scenario. Through enumeration, permutation and combination of scenarios, SADPBA developers check and validate the design results and create deterministic traceable relationships.

- In the same space, elements that have dependency relationship have traceable relationship.
- When DSF is complete to DFA, the elements in DSF and in DSA have traceable relationship.
- When DSF is complete to DSs, the elements in DSA and in DSs have traceable relationship.

IV. STUDY CASE MEECS

In this section, we present the Mobile Embedded E-Commerce System (MEECS), by which we perform the explorative research on applying agent into mobile e-commerce. In the development of this system, we introduce and refine SADPBA discussed above. In the following content, we first give an introduction to this system and then explain how SADPBA was used parts.

A. Applying SADPBA in MEECS

In this section, we briefly introduce the design process based on SADPBA, and describe how the decisions about architecture of MEECS are made by comprehensive consideration of requirements and other concerns such as techniques and research related. We start from the enumeration of primary requirements of our project. Just like what most people are doing nowadays, we use the artifact "use cases" document to record our project's goals. First, we categorize three kinds of actors in this system, the client users who use the handheld mobile devices, the administrators of Agent Router, and the vendors who provide services.

B. Requirements Analysis

The goal of client users is very simple: they should see the result generated by the service through user interface and can send requests to find and choose some services. The heterogeneity of display in various devices should be taken into account, which triggers our another research project, the Language facilitating Interface Representation under Limited Mobile Computing Environment (FIML) (Wang, 2003). Simply speaking, this is a markup language specifically for graphical display in different mobile devices. It is rather trivial to talk more about this language in this section. What you should know is that this language needs parsing and behaviors represented by it should be executed. Therefore, the use case diagram seems like what is shown in below Fig. 3.

The tasks of administrators are to maintain the Agent Router's map records, each of which bind a client user to a service. For this, we introduce the idea of administrator agent, an independent component capable of performing administrators' duties, such as registering or unregistering a client user, finding the suitable service brokers and finally clearing the channel enabling communication between client user and service directly. Other than the functional requirements, the performance and availability concerns need to find their solution here. In the architecture design of this part, you can see that how these requirements are met. The vendors have fewer responsibilities. For this system, they
only need to implement their services by following a specification which guarantees that their services can be recognized and used by the agents. Since it is relative easy and not the key point of MEECS, we postpone its design until the late phase of system. Generalizing the points above, we generate three associated packages of use cases as the start point to the architecture design phase.

C. Architecture Design

In the architecture design phase, we separate the whole system into three sub-ones. The client, which is called as "terminal component" in the architecture design, focuses on the representation functions. And the Agent Router is designed as Mobile Embedded E-Commerce Platform (MEECP).

- **Terminal:** The architecture of terminal is shown in Fig. Terminal contains the components listed as following:
- **Facade:** Responsible of construct and display of user interface.
- **Command:** Parsing the behavior mark in the FIML interface, generating the command objects and then performing their execution.
- **Parser:** Parsing the mark of FIML and extracting the data embedded in them, preparing for further operations such as command execution or interface display.

![Fig. 4. Architecture of MEECS terminal](image)

- **RMS:** Archiving the data permanently in the mobile terminal, enabling the operations of records query, insert, delete and-update. RMS (Record Management Storage) originally is the facility provided by J2ME for permanent record storage. We extend it to make convenient the operations of records designed in our own system.

- **MsgCenter:** Under the mobile environment, network breaking off is the conventional problem. In this concern, MsgCenter is designed to work in the mail box style, or more canonical, the asynchronous transmission, allowing connection interrupt and resume. MsgCenter is the sole port that links the terminal distributed discretely in the net. The architecture for client supports three main processes: the generation of flexible user interface, the execution of behavior and the messages handling. The user interface is generated in the following steps. The FIML Parser component parses the control description in the markup language, and then extends the variables, replacing the configuration table, the system parameters and page parameters. The Facade component gets the parse result, a description of controls, and finally renders the display area. Execution of behaviors starts from the interaction from users. Once people trigger some behavior, its description is fed into the Command component, which first replaces the parameters in the description and extracts the elements of that behavior.

![Fig. 5. Architecture of MEECP](image)

To solve the problem of task allocation and the efficiency of communications among agents, MEECP introduces the Agent Layered Management Architecture. The agents are separated into five layers according to their duties. The Layer of Common Utility, supported by Agent Management System and Directory Function, facilitating the basic agent information keeping, including the identifier, status, time stamp and bound services. The messages coming from outside will be handled by this layer. Therefore, administrators of MEECP can impose policies by filtering out the messages needing to ignorance. The Layer of Management control all the agents deployed on the platform. And noticeably, it is itself also an agent, called Super Server Agent. In the perspective of functions, Super Server Agent does nothing about what the terminal wants, but is the start point of task allocation.

The Layer of Inception finishes the job of connecting one terminal to a carefully selected Function Agent, the proxy of actual services. In this layer, two kinds of agents, Administrator Agent and Server Agent cooperate to balance the load of numerous agents.

The Layer of Service Category is set for category of services. It is this layer that keeps the information of a set of
related services registered and prepares to choose one for the
terminal according to the guide tips attached by it in the form of
properties.

The Layer of Domain Specific Services provides the final
service proxy. After all steps of registering, the terminal and
its bound services are linked by this kind of agents. Of course,
one can realize the function in this agent so that there is no
burden of service objects. However, doing this too much will
exceedly increase the load of MEECP and affects the
consistency. System Design It is time to convert architecture
to the model that is easily to be
implemented-related models, where we use UML to
describe the domain specific concepts during the
requirements analysis. Those concepts related to the solution
of problem require much more attention. The result of this
phase is so called domain model, which although cannot
solve the problem but provide an order-of-magnitude sense
to aid developers to gain a deeper insight.

In the case of MEECS, the concepts such as mobile client,
agent, register information keeper, are identified to define the
requirements. The requirement analysis is in fact an action
that finding the constraints and conditions the entities in the
domain model have to conform to. Note that the object model
is neither a description of architecture nor class or objects in
the Object-Oriented development, but only a simplification
and visualization of concepts in the problem world. The
architecture design is the first step getting approach to the
solution in the high level of the system. Although algorithms
are so important to succeed, they only deal with the problems
about computation, which is only a small part in the usable
software. More broadly, requirements of quality attribute,
performance, usability, availability, security and so on, have
to be tackled by the cooperation of decomposed elements. In
the architecture design, concepts are separated according to
the requirements into elements and regulate their interaction.
In MEECP, for example, in order to balance the load, we
design the layered agents. Abstractly, this is a map from
function design space to the architecture design one, which if
gotten verified again and again we can put in the rules of map
created for this domain, and then contributes to the automatic
design. Architecture design is programming language neutral
in which only the overall sight of final system is generated.
Finally in the system design, we determine the techniques
and convert the architecture into the model that is easily to be
coded.

For instance, we use RMS provided by J2ME to achieve
the permanent information storage. We convert ports in the
architecture to the methods enclosed by Java interface. And
we implement various agents into a tree of generalized
hierarchical agent classes.

V. CONCLUSION

We extend the design space by splitting it into function
design space, architecture design space and system design
space, which is the core idea of SADPBA, an
architecture-centered design process. We develop MEECS, a system of mobile e-commerce, in the design phase of which we use SADPBA. In this case, three design phases are listed in more detail to illustrate how an element of one design space is mapped.

REFERENCES


Dr. D.Vasumathi received the B. Tech. from JNT University, Hyderabad, M. Tech degree in Computer Science and Engineering from JNT University, Hyderabad and Ph. D in Computer Science and Engineering from JNT University, Hyderabad, Andhra Pradesh, India. He worked as Assistant Professor, Presently working as Associate Professor in JNT University, Hyderabad. He has 12 years of Teaching experience. Currently guiding 8 Ph. D scholars. He is life member of ISTE and IE. Her research interests Data Mining, Image processing, Web technologies. He has 5 international publications and 10 international conference Publications.

P. Rajarajeswari received the B. Tech. from S.V University, Sri Venkateswara University, Tirupathi, Andhra Pradesh. M. Tech degree in Computer Science from JNT University, Hyderabad and Pursuing Ph.D in Computer Science and Engineering, JNT University Hyderabad Andhra Pradesh, India. Presently she is working as Assistant Professor of Computer Science and Engineering in Madanapalle Institute of Technology and Science, Madanapalle, Chittoor(Dt), Andhara Pradesh. He has 8 years of and teaching experiences. She is a life member of IAENG and IACSIT. Her research interests include Software Architecture, Software Engineering and Data Mining. He has 5 international publications and 4 international conference Publications.

Dr. A. Rama Mohan Reddy received the B. Tech. from JNT University, Hyderabad in 1986, M. Tech degree in Computer Science from National Institute of Technology in 2000 Warangal and Ph. D in Computer Science and Engineering in 2008 from Sri Venkateswara University, Tirupathi, Andhra Pradesh, India. He worked as Assistant Professor, Associate Professor and Presently working as Professor of Computer Science and Engineering, Sri Venkateswara University College of Engineering. He has 28 years of Industry and Teaching experience. Currently guiding twelve Ph. D scholars. He is life member of ISTE and IE. His research interests include Software Architecture, Software Engineering and Data Mining. He has 10 international publications and 14 international conference Publications.