Mei H, Liu XZ. Internetware: An emerging software paradigm for Internet computing. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 26(4): 588–599 July 2011. DOI 10.1007/s11390-011-1159-y

Internetware: An Emerging Software Paradigm for Internet Computing

Hong Mei (梅 宏), *Fellow*, *CCF*, and Xuan-Zhe Liu (刘譞哲), *Member*, *CCF*

Key Laboratory of High Confidence Software Technologies (Peking University), Ministry of Education, Beijing 100871, China
School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China

E-mail: meih@pku.edu.cn; liuxzh@sei.pku.edu.cn

Received February 26, 2011; revised May 23, 2011.

Abstract The Internet is undergoing a tremendous change towards the globalized computing environment. Due to the open, dynamic and uncontrollable natures of the Internet, software running in the Internet computing environment has some new features, which bring challenges to current software technologies in terms of software model, software operating platform, software engineering approaches and software quality. Researchers in China have proposed the term "Internetware" to present the emerging software paradigm. Sponsored by the National Basic Research 973 Program, several research practices have been done on the Internetware in the past decade. This paper summarizes the progress and status of the Internetware researches. A technical solution framework for the Internetware paradigm is proposed from four aspects: the Internetware software model defines *what the Internetware is to be*; the Internetware middleware determines *how to run the Internetware applications*; the engineering methodology determines *how to develop the Internetware applications*; the Internetware quality assurance determines *how well the Internetware applications can perform*. The paper also discusses the ongoing research issues and future trends of Internetware.

Keywords software engineering, Internetware, Internet computing

1 Introduction

Software is a computer program that models the problem space of the real world as well as its solution. Software paradigm (or program paradigm) describes a software model and its construction theory from the perspective of software engineers or programmers[1]. As shown in Fig.1, a software paradigm usually concerns four aspects: *what is to be* constructed and executed (software or program model); *how to develop* the resulted software artifacts or entities (development techniques, e.g., programming languages, engineering approaches and supporting tools); *how to run* the artifacts or entities (runtime system supports, such as the operating systems or middleware platforms); and *how well* the constructed and executed software can perform (the promised software qualities, e.g., correctness, performance, reliability, user experiences, and their assurance mechanisms via construction and operation).

With the rapid development of computer science and technology, the application domain and runtime environment evolve as well. As shown in Fig.2, the computing application domain keeps expanding, from science computing to enterprise/personal computing and then to "anytime-anywhere" ubiquitous computing. The runtime environment is evolving from the single-machine to the networked and current the Internet environment. With the evolution of application domains and runtime environments, the scale and complexity of software systems keep increasing. Powerful software technology is the key to meet the ever-increasing application demands for high productivity and quality supports. Software paradigm acts as the core driven force the evolution of software technology. Software paradigm always pursues to better utilize underlying hardware capabilities or runtime features. It attempts

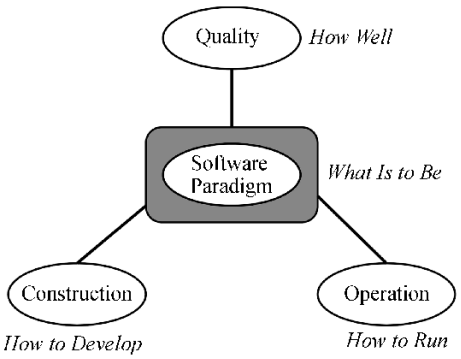


Fig.1. Software paradigm.

Survey

This effort is sponsored by the National Basic Research 973 Program of China under Grant No. 2009CB320700, and the National Natural Science Foundation of China under Grant No. 60821003.

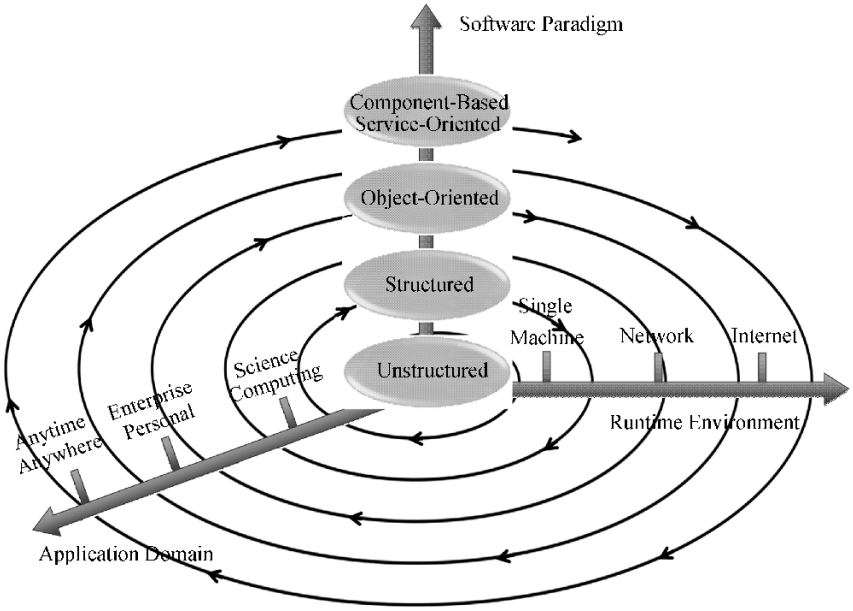


Fig.2. Evolution of software paradigm.

to provide a more expressive and natural computing model from the application perspective. In the history of computer science and technology, there have emerged some software paradigms, including the structured paradigm, the object-oriented paradigm, the component-based paradigm and currently the popular service-oriented paradigm. One observed fact is that, when the software paradigm evolves, revolutionary challenges and tremendous opportunities come to software technology as well.

For every paradigm, we might find that, the *WHAT-IS-TO-BE*, the *HOW-TO-DEVELOP* and the *HOW-TO-RUN* issues are always prior to be solved in 5∼10 years since the paradigm emerged, and then we come to the *HOW-WELL* issue. For example, we can review the history of the object-oriented (OO) paradigm. The object-oriented software model attempts to natively describe the structure of the objective world, such as the abstraction and encapsulation of objects/classes, and its relationships such as inheritance. Since the 1980s, many programming languages emerged to support the OO program development, for example, Smalltalk and C++ were born in the 1980s, while Java became popular in the 1990s. When OO systems were widely applied in the 1990s, more attentions were paid to their high-quality. For example, OO analysis can identify the requirements with use case models; OO design can elaborate the analysis model to produce design specifications; OO testing techniques can verify the correctness and consistency of the OOA/OOD models as well as the implementation.

The fast development of the Internet has made itself perform as a globally ubiquitous infrastructure. Moreover, beyond its basic connection capabilities, the Internet is now growing up into a "*Global and Ubiquitous Computer*". The present Internet not only consists of a huge and ever-growing number of diverse computing devices, but also serves as a platform with much more powerful supports for problem solving. The Internet computing environment distinguishes from the traditional computer systems, with several new features, such as the distribution and decentralization of control, the diversity of network connections, the openness and changes of network environments, the autonomy of computing nodes, the heterogeneity of human, devices and software, the personalization of usage patterns.

The "Internet-as-a-Computer" has significant impacts on computer science and technology. Some new concepts, application modes and technologies have emerged, such as the popular Grid Computing, Cloud Computing, Services Computing, Pervasive Computing, Semantic Web, Web 2.0, Social Networking, Smarter Planet. These new concepts, application modes or technologies view the "Internet-as-a-Computer" from different perspectives. From the perspective of resource sharing, Grid Computing aims at the transparent aggregation of resources from network nodes. From the perspective of resource management and on-demand delivery, Cloud Computing aims at the concentration and virtualization of resources over data centers. From the perspective of human-computer interaction, Pervasive Computing aims at making future network applications operated and accessed ubiquitously. From the perspective of "Everything as a

Service", Services Computing aims at the coordination and dynamism of services.

From the perspective of software paradigm, we try to investigate the software running in the Internet computing environment. Unlike traditional software systems running on local single machines, software systems on the Internet might consist of several distributed and autonomous software entities. These entities can collaborate with one another in various fashions. In this way, a "*software Web*" emerges, which is similar to the WWW resulted from Web pages and hyperlinks among them. Like the WWW offering information services such as information publishing and sharing, the software Web provides plentiful services of various functionalities online for the users. The software Web is able to be aware of the dynamic changes of the contexts, and then continuously evolves with functionality and quality-of-service.

Due to the open, dynamic and ever-changing natures of the Internet computing environment, software on the Internet has some new features compared with the traditional software. Take software collaborations for example. They are always realized by means of interactions among software entities. In the object-oriented paradigm, both the objects and their interactions are fixed, because one class/object has to explicitly declare the invocation of another class/object. In the component-based paradigm, component collaborations are more *loosely-coupled*. For example, Java Enterprise Edition (Java EE) can support the dynamic lookup and binding of proper components (using Java Naming and Directory Interface). Although the interactions are still fixed, the interacted components can be dynamically changed. In the service-oriented paradigm, service collaboration becomes much more flexible. Different services can interact with one another in different flows to accomplish the same goal. For instance, in WS-BPEL, it is allowed to compose different function-identical candidate services, and their interaction styles might vary as well.

However, in the Internet computing environment, more and more goals are unclear, undetermined or even undesired before collaboration finally performs. In other words, these collaborations are "*emergent*" rather than predefined. Such emergent collaborations may occur in many scenarios, for example, in the prevention and control of epidemic (such as SARS and H1N1). Under normal circumstances, the Chinese Center for Disease Control and Prevention (CDC) might not have fixed relationships with other organizations (such as airport, hotel and hospital). However, some emergencies (such as H1N1) will result in the on-demand collaborations of these organizations. One possible scenario can be described in Fig.3. When a passenger is thought to be infected with H1N1 a message will be sent to the early-warning system of CDC. CDC will immediately request the airport management system for passenger list on the same flight. Then CDC will contact the hotel to arrange rooms for isolating all the passengers, and notify the hospital to take cases of the "infected" passengers in the hotel. In this scenario, accomplishing the emergent goal requires powerful and flexible technologies to support on-demand collaborations among these organizations.

Compared with the features supported by existing software paradigms, the features for software on the

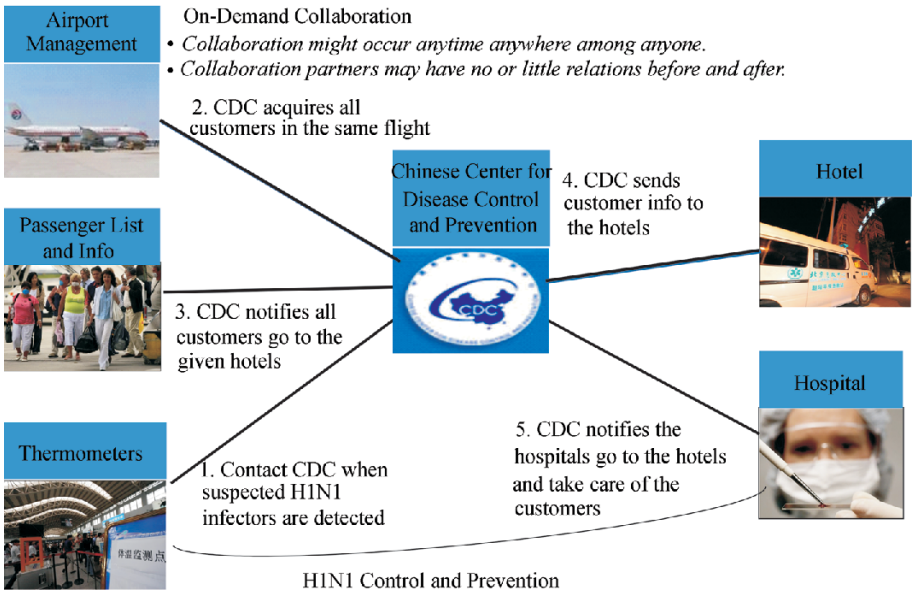


Fig.3. On-demand collaboration scenario.

Internet might not be totally new. However, some improvements should be considered. We argue that a new software paradigm for the Internet computing environment is required. It shall systematically support the development, operation and quality assurance for software on the Internet.

Researchers in China proposed the "*Internetware*"[1-3], which literally means "*the Software paradigm for the Internet*". Sponsored by the National Basic Research 973 Program of China, systematic efforts have been made on Internetware research and practices. In the past decade, two 973 projects has been carried out consecutively: "*Research on Theory and Methodology of Agent-Based Middleware on Internet Platform*" (2002∼2008), and "*Research on Networked Complex Software: Quality and Confidence Assurance, Development Method, and Runtime Mechanism*" (2009∼present). About 80 researchers from Chinese universities and institutes (including Peking University, Nanjing University, Tsinghua University, Institute of Software of the Chinese Academy Sciences, the Academy of Mathematics and Systems Science of Chinese Academy Sciences, East China Normal University, and IBM China Research Laboratory) have participated in the projects.

This paper will present the concept and principles of Internetware, and introduce the progress and current status of Internetware research. The paper is organized as follows. Section 2 describes the Internetware paradigm and states the key challenges. Section 3 presents the architecture-centric technical solution framework of Internetware from three aspects: Internetware model, Internetware operating platform, and Internetware engineering approach. Section 4 discusses future research. Section 5 ends the paper with conclusion remarks.

2 Internetware: A New Software Paradigm

2.1 Basic Concepts of Internetware

Essentially, Internetware is constructed by a set of autonomous software entities distributed over the Internet, together with a set of connectors enabling the collaboration among these entities in various fashions. The Internetware software entities are able to be aware of the dynamic changes of the running environments, and continuously adapt to these changes by means of structural and behavioral evolutions.

As shown in Fig.4, from the micro perspective, Internetware software entities collaborate with one another on demand. From the macro perspective, the entities can self-organize an application domain or community-of-interest[3]. As a result, the development of Internetware can be viewed as the continuous and iterative composition of various "*disordered*" resources into "*ordered*" software systems. This implies that the Internetware development is a bottom-up, inside-out and spiral process.

Obviously, Internetware distinguishes from traditional software in terms of forms, structures and behaviors. The Internetware applications should be autonomous, cooperative, situational, evolvable, emergent and trustworthy[1].

• *Autonomous.* It means that Internetware entities are usually distributed, relatively self-contained and independent. They usually perform according to the composition or deployment strategies defined by their providers, and continuously satisfy the providers' goals. Internetware can adapt itself when necessary, by sensing and collecting information of environment changes. For example, in the above scenario, the CDC, airport,

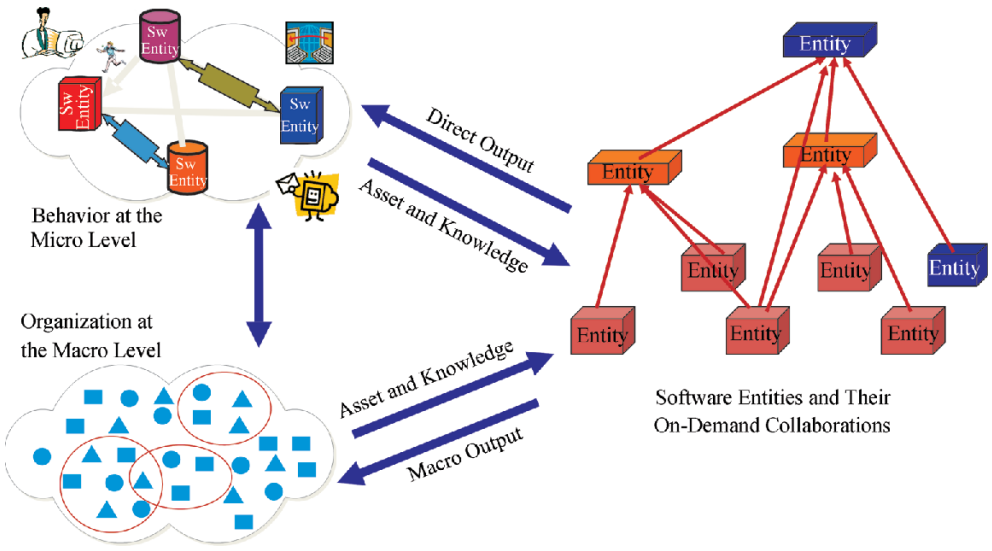


Fig.4. Behavior, organization and composition modes of Internetware.

hotel and hospital are all autonomous entities, and perform on-demand collaborations with one another.

• *Cooperative*. It means that a set of Internetware entities might collaborate with one another, and constitute new Internetware applications. Most of the time, the collaborations are rather dynamic than static to adapt to the user requirements and environments on-demand. The collaboration mechanisms between Internetware entities may be various, and can be changed if necessary.

• *Situational*. It means that Internetware systems can be aware of the runtime contexts and scenarios, including underlying devices, operating platforms, networking conditions, application contexts, or the changes of other Internetware, etc. Hence, both the Internetware entities and their operating platforms might be capable of exposing their runtime states and behaviors in some way.

• *Evolvable*. It means that the structures and behaviors of Internetware applications might dynamically change. The Interntware applications usually consist of autonomous entities over the Internet, and provide online and continuous (e.g., 24 hours $\times$7 days) services for a number of users. Hence, the Internetware applications cannot be completely shut down. Internetware applications have to perform online evolutions to accommodate the user requirements and the environments. Possible evolutions may include the adding/removal of entities, the changes of functionalities on-the-fly and just-in-time, the interaction styles between entities, the change of system topologies, etc.

• *Emergent*. It means that Internetware systems have un-designed behaviors in or undesired effects on the runtime instances or interactions and they might iteratively result in more and more changes of system structure and behavior. In this way, the Internetware systems have some typical features of complex systems.

• *Trustworthy*. It means that Internetware systems should promise the comprehensive tradeoffs among several quality attributes. As Internetware applications online serve a number of users, the trustworthiness of Internetware might cover a wide spectrum, with properties such as reliability, security, performance, user experience, etc. Quality assurance might be relevant to several aspects, including the autonomous entities, the interaction styles, the network environments, the usage patterns, the malicious attacks, and the effects caused by system evolution, etc.

2.2 Challenges Statement for Internetware

Considering the four aspects of software paradigm shown in Fig.1, the Internetware paradigm has to address the following technical challenges:

• *Software Model* (*what is to be*). The Internetware model should specify the form, structure and behavior of the software entity as well as the collaborations. These will determine the principles and features of the corresponding software technologies (programming languages, development approaches and runtime mechanisms). The Internetware software model should leverage both legacy software and new features. The basic Internetware entities can be built upon current popular technologies, such as object-oriented technologies or services computing technologies. But new capabilities should be provided, for example, the diverse and flexible connectors to enable on-demand collaborations among entities, the context-aware and situation-aware capabilities.

• *Software Operating Platform* (how to run). It realizes the elements and their relationships of the software model. Internetware operating platform should provide a runtime space to operate Internetware entities and their collaborations. It should equip legacy software systems with Internetware features, and should also manage the applications and itself in a more intelligent and automatic manner.

• *Engineering Approach* (*how to develop*). The engineering approach should systematically control the whole lifecycle of Internetware development, including requirements specification, design, implementation, deployment, maintenance and evolution. The Internetware engineering approach should identify self-organized communities and application domains, facilitate the self-organizations, and involve all stakeholders, especially the actual end users.

• *Quality Assurance* (*how well software systems perform*). Software systems on the Internet usually serve a large number of users in an online and simultaneous style[4]. Internetware quality framework should not only define quantitative and qualitative measurement methods for various quality attributes such as performance, reliability and usability, but also make comprehensive tradeoffs among these attributes. To promise the Internetware quality, it requires the quality assurance mechanisms by both engineering approach at development time (e.g., testing, verification and validation) and middleware at runtime (e.g., online evolution, autonomic system management).

3 Technical Solution Framework for Internetware

Realizing the above challenges, the researchers in China have made some attempts and efforts in terms of theories, models, approaches, technologies and applications for Internetware paradigm. The results can be concluded as an "*architecture-centric*" technical

solution framework[5], which covering three aspects.

• The Internetware software model concerns three aspects: entities, collaborations and environments, as well as their relationships. An Internetware entity should have basic business functionality interfaces to enable the collaborations. It should also have the capabilities to expose its own states and behaviors. Facilitated by context modeling, the environment information should be monitored and captured by the entities. Governed by the software architecture model, the collaborations among entities can be also globally planned and adapted.

• The Internetware middleware takes charge of operating Internetware applications. In the middleware, the container takes charge of realizing the Internetware entities, and provides some advanced capabilities and services. For example, the flexible connectors can mediate different protocols among entities; the dynamic binding of policies can satisfy the constraints. Runtime Software Architectures (RSA) is employed to govern the on-demand collaborations. By leveraging autonomic computing for management, the Internetware middleware supports the self-organization and self-adaptation of Internetware applications, and promises high quality-of-services (such as reliability, performance) at runtime. Particularly, the structure of Internetware middleware should be open and expandable, so new capabilities and services can be on-demand loaded or customized.

• The Internetware engineering approach follows the core principle of "*Software Architecture of the Whole Lifecycle*". Software architecture acts as the blueprint and controls every stage of Internetware development. To support the online development of Internetware applications, the Internetware entities and their on-demand collaborations are implemented and governed based on software architecture. To better control the development process, domain modeling techniques are employed to organize heterogeneously distributed resources of a specific domain.

In the remainder of this section, we will introduce the efforts and results of the three aspects respectively.

3.1 Internetware Software Model

To support the features of Internetware, the Internetware software model mainly focuses on entities, collaborations and context-awareness[5].

First of all, Internetware entities should be in the form of software components, so they can be independently deployed. They should have not only business functionality interfaces for computation, control and connection, but also some new advanced capabilities, e.g., the reflective interfaces to expose the states and behaviors, and monitor the environments as well. Moreover, they can employ some advanced facilities (such as rule engines) to plan and adapt their own business functions based on reflection.

Secondly, in the Internet computing environment, software entities might cooperate with one another in various manners. The collaborations cannot be predefined, and might be emergent at runtime. It requires the separation of collaboration logics from software entities.

Thirdly, in the Internet computing environment, the software entities may be added, removed or changed anytime, whereby the interactions of software entities are more flexible and changeable. The Internetware software model has to pay careful attention to contextual factors from several aspects, such as the network environments, user preferences, and collaborative entities, etc.

Addressing these issues, several efforts have been made on Internetware software model. The researchers in Peking University proposed the autonomous component model[6-10]. Such a model is built upon the current software component technologies, including EJB components or Web services components. The entities include the component meta-model and reflective interfaces for adapting states and behaviors, and can further leverage the environment meta-model and reflective interfaces to capture the context information. The entities with rule engine can reason out the proper actions based on the context information and some rules, so that the adaptation can be done more intelligently and automatically.

The researchers in Nanjing University leverage the software agent and service computing technologies for the Internetware software model[11-13]. They realize the software entities as agent-based software services, and model the environment using ontology techniques. The components can perform multi-mode interactions, capture the user-system-environment interaction behaviors, and reason the adaptation based on environment ontology.

3.2 Internetware Operating Platform

The operating platform usually provides a set of necessary infrastructural services for running application software[2]. In the network computing environment, operating platform has to provide more capabilities, e.g., supporting different interoperability protocols, heterogeneous components collaboration, network resources aggregation and management, etc. Middleware is a special kind of system software to enable the interactions among distributed application software with common services support (e.g., interoperability)[14]. Middleware technologies, such as J2EE/EJB, CORBA/CCM and

.NET/COM, have been widely adopted in enterprise computing environment.

However, current middleware cannot well support applications in the Internet computing environment[15]. To support the new requirements of Internetware, some Internetware middleware prototypes have been developed based on some popular technologies. For example, the PKUAS[16] and OnceAS[17] are both Java EE compliant application servers; Artemis-* is targeted at the software agents[11-12]; and OnceSE[18] is implemented to support Web services. The Internetware middleware supports Internetware applications in four aspects[5]: 1) the incarnation of Internetware entities; 2) the support for on-demand collaborations; 3) the autonomic management of Internetware middleware; and 4) the extensibility of Internetware middleware.

3.2.1 Incarnation of Internetware Entities

The basic functionality of Internetware middleware is to realize Internetware entities, either based on currently popular technologies (e.g., Java EE and Web services) or for new technologies (e.g., software agent). The Internetware middleware has to provide the container with necessary capabilities. For example, for an autonomous component, PKUAS allows a normal Java class which implements some business functions to be deployed into its container. PKUAS provides an open framework to facilitate heterogeneous components with different interoperability protocols, e.g., RMI, SOAP and HTTP[16]. It also allows flexible binding of various facilities such as security policies, the database connections or even the intelligent facilities for rule-based component self-adaptation[6]. Furthermore, PKUAS allows dynamic add/removal of components[19] and upgrade/degrade of connectors[20], and supports the online evolution of Internetware applications[21].

3.2.2 Supports for On-Demand Collaborations

The Internetware middleware has to support the on-demand collaborations of Internetware entities. To systematically control the collaborations from a global and comprehensive view[10], Internetware middleware employs the Runtime Software Architecture (RSA) and reflection mechanisms[22] in several prototypes (including PKUAS, JOnAS① and Apache Axis②). RSA takes the causal connections of the running system, and benefits from the reflective middleware's deep control of the whole system. Such a causal connection ensures that all the changes of RSA can immediately result in corresponding changes of the running system, or vice versa. The collaborations between entities are then fully under the governance of RSA.

3.2.3 Autonomic Management of Internetware Middleware

As Internetware middleware is responsible for supporting the applications online and promising qualities, it has to manage the applications and itself more intelligently and automatically[15-16]. The management of Internetware middleware can be based on the classical autonomic computing MAPE (monitor-analyze-plan-execute) loop. The autonomic management of Internetware middleware should consider the scope, operability and trustworthiness. Based on RSA, an architecture-based and model-driven self-adaption technique, called SM@RT (it means "*software model at runtime*"), is proposed for the autonomic management[23-25]. In SM@RT, the software architecture of the middleware as well as its applications defines the scope and connotation of middleware management. With the RSA-based monitor and control, the correctness and effectiveness are ensured through software architecture analysis and evaluation[24]. Addressing key challenges of autonomic computing (knowledge representation, analysis and decision making), the formal descriptions of architectural decisions represent the knowledge. Architecture styles and patterns, domain-specific software architectures and application architectures specify the common-sense, domain-specific and application-specific knowledge, respectively. The architecture descriptions make knowledge representation more well-structured and easy-to-interpret. Either the SA dynamics by imperative descriptions or the self-adaptive SA by declarative descriptions can be used to plan adaptations[25]. In other words, middleware can automatically derive the adaptation plan based on the knowledge, e.g., using rules like ⟨*event, condition, action*⟩ to derive the proper actions for the given event and condition. The actions might re-factor partial software architecture, by interpreting the knowledge derived from the descriptions of bad patterns and the corresponding good patterns. Finally, new styles can be merged with the existing ones, to make SA of the whole systems perform as desired.

3.2.4 Extensibility of Internetware Middleware

All the capabilities of Internetware middleware require the middleware be well-structured, flexible and extensible. For example, PKUAS is completely component-based with a micro-kernel-based structure[16]. The middleware capabilities, including

① JOnAS is an open-source JEE application server developed by OW2.

② Apache Axis is an open-source Web services engine developed by Apache.

those containers, protocols, services, facilities, mechanisms and frameworks, are all encapsulated into components. The micro kernel takes charge of the customization, extension and autonomic management of these platform components. With the open structure, new advanced capabilities can be dynamically added. For example, PKUAS allows the fault-tolerant service[21] to be dynamically plugged in-and-out; OnceAS allows customization of the data persistence service, the constraint services for the access control[17], and the relaxed transactions processing service at runtime[18]; Artemis-* supports the reliable messaging service[26] and integrates the trust computing features for reputation ranking[13], etc.

3.3 Internetware Engineering Approach

Software development methodology determines the directions and principles of software development process. It always aims at improving the development efficiency and quality.

For Internetware development, a software architecture centric engineering approach is proposed. The engineering approach leverages the ABC (architecture-based component composition) methodology[7]. ABC employs software architectures as the blueprints to develop software systems. It aims at composing reusable components under software architecture, and shortens the gaps between high-level design and implementation with supporting tools and mapping mechanisms. The Internetware engineering approach supports the development from three aspects.

3.3.1 Software Architecture of the Whole Lifecycle

The guiding principle of Internetware development methodology is "*Software Architecture of the Whole Lifecycle*"[7]. It means that, software architecture acts as a blueprint and plays the central role in every phase of Internetware lifecycle, including requirement analysis, design, composition, deployment and maintenance. In other words, software architectures govern the development at a systematic and global level.

The requirement analysis of Internetware outlines the concepts of software architecture. In the Internetware development, the requirement analysis artifact should be transformed into software architecture more conveniently, naturally, and directly. There have been some architecture-oriented requirement analysis approaches proposed, which are used directly to obtain the conceptual software architecture[27-29]. For example, the feature model from feature-oriented requirement analysis[30] can facilitate the generation of the conceptual software architecture semi-automatically.

The design of Internetware leverages the design view of software architecture. Based on the requirement specifications of the conceptual architecture, designers can make architectural design decisions[31], refine the candidate components and connectors. Finally, the static and dynamic software architecture models will be constructed. Designers or the development tools should maintain the traceability between the requirement specifications and design models[32]. The Internetware design should reuse any existing reusable assets, e.g., the components retrieved from component libraries[33], patterns captured from open source software[34], or collective-intelligence derived from online community services[35].

After the design phase, the overall system architecture is obtained. The basic functional units of Internetware are the existing and running entities. Correspondingly, the main implementation task of Internetware development is not programming, but composition of these entities[36]. The composition of Internetware systems includes discovery and selection of candidate entities based on the software architecture design. The collaboration of the entities should follow the architectural specifications as well. The composition can be either automatic (e.g., by means of agent-based[8] or model-driven techniques[37]), semi-automatic (e.g., by means of assistance to compatible components[38]) or totally manual. When the entities or collaborations do not fit the software architecture, necessary adaptations are required.

The deployment of Internetware usually involves a number of items that require manual configurations by the deployers. Most of the information can be directly obtained or transformed from the design and composition stages[23]. In some cases, new emerging entities as well as the interactions might require changing the partial or whole organization of existing entities, which lead to the tedious and trivial task. Therefore, the deployment view of software architecture is used to maintain information coming from the design and implementation[22].

Internetware development is usually an iterative refinement. It maintains the mapping and transformation of software architectures of the real target system. In the stage of maintenance and evolution, the runtime view retrieves the actual runtime states of the system. Thus, it provides the most accurate and complete information. In current Internetware solution framework, the RSA usually cooperates with reflective middleware, captures the runtime information, makes analysis, and performs the online maintenance and evolution without stopping the whole running system[19].

3.3.2 Implementation of Internetware Entities and Their Collaborations

Naturally, the development of Internetware entities is actually to develop new software systems with Internetware features, or enhance existing traditional software systems to have Internetware features[5]. Currently, both the types can be supported and governed by the software architecture (SA).

Take the Internetware adaptation for example. Firstly, SA models are used to analyze the qualities of Internetware adaptation for locating the elements to be adapted. Then, SA models with dynamic capabilities record what should be done at runtime to achieve the desired qualities. Finally, the designed adaptation is executed by RSA without stopping the running system. The RSA can automatically detect if the running system has ill structures, and dynamically refactor them into good structures. Particularly, reasoning rules can also be integrated into SA, and enable dynamic adaptation with rule engines.

The on-demand collaborations of Internetware entities are still controlled by SA. Beyond Internetware entities development, the development of on-demand collaborations should take into account of the features of both entities (e.g., distribution, autonomy and heterogeneity), and the entity interactions (e.g., diversity, complexity and changeability).

3.3.3 Domain Modeling for Internetware

Domain engineering is a promising way to address the creation of domain models and architectures and the identification of reusable assets. In a sense, the domain analysis is a process in a bottom-up fashion, which coincides with the engineering of Internetware.

In the Internetware engineering approach, the disordered resources over Internet are specified into domain model, which represents high-level business goals of a set of Internetware systems. Requirement specifications are generated for a new application by tailoring and extending the domain model. Finally, new entities and collaborations are implemented. When time elapses, the new application may be scattered somewhere on the Internet as a service, and becomes a new disordered resources. In turn, these new disordered resources can be added into the further domain model. For the purpose of knowledge-driven domain analysis, ontology can facilitate to realize the (semi-)automated identification of concepts as well as their relationships[27], and consolidate the knowledge into more representative forms[39-40].

3.4 Effects and Impacts of Internetware

In summary, several important and valuable results in the Internetware research have been obtained in the past ten years, and in academia, over 100 high-quality referred papers have been published in very influential international journals and top conferences.

Meanwhile, Internetware has gradually shown its research potential. For example, IBM Global Technology Outlook (GTO)③ adopted some of the ideas and principles of Internetware in its 2010 GTO topic "*Future Software Technology Trends*". GTO regards that the software paradigm for "*Internet as a Computer*" does reveal a new perspective on software technology in the future. Some results have been open-sourced on famous communities, for example, some capabilities of Internetware middleware, e.g., online evolution and dynamic clustering, are now adopted by the Jon^{2}AS application server, and open-sourced in OW2 community. Since 2009, a special international symposium for Internetware research has been held, with attendees from China, USA, Europe, Australia, Japan and Korea. More and more attentions have been paid to the Internetware research, from both domestic and international counterparts.

4 Ongoing Research Work

Based on the existing research results, some new Internetware research issues are in progress to accommodate recent trends of network computing environment.

Firstly, heterogeneous networks are converging. Telecom network, mobile network, sensor network and other ad hoc networks are now capable of connecting and collaborating with the Internet. This results in the hybrid complex network environment. The Internet will play the central role of connecting the IT world, physical world and human world as a new complex "cyberspace". Computing devices, human society and physical objects will be seamlessly integrated together. In such a cyberspace, software systems will orchestrate the information, process, decision and interactions among the human world, IT world and physical world. In other words, the cyberspace will be "software-intensive"[41]. Correspondingly, the software systems in the cyberspace might become larger in scale and more complicated.

Secondly, the cyberspace significantly extends the application width and depth，and the focus on software

③IBM GTO is an annual Information Technology report, accomplished by more than 3 000 researchers world wide. GTO forecasts future trends in IT. It is famous due to accurate and deep analysis as well as leading some important IT evolutions, such as e-Business, SOA and Smarter Planet.

quality shifts as well. From ISO-9126 definition, software quality consists of the internal quality (attributes which do not rely on software execution and can be statically measured); external quality (attributes which are applicable to running software); and the quality in use (attributes which are only available when the final product is used in real conditions). The internal quality determines the external quality, and the external quality determines the quality in use. Internal and external qualities are relatively objective, since they can be guaranteed by developers or constrained by the agreements between users and developers. However, in the cyberspace, software systems directly serve millions of users with various online services. The diversity of network environments, devices, and user preferences make the quality assurance more challenging and complex. Internetware systems should be of "high-confidence"[14], which is more "*user-centric and user-oriented*". In other words, more attentions should be paid to the quality in use. Quality in use will be relatively subjective, since it significantly relies on users' individual preferences, experiences, and ranks. For assuring the quality in use, based on the assurance mechanisms of system quality, high-confidence has to comprehensively measure a set of quality attributes as well as their tradeoffs.

These two trends lead to some new research issues of software technologies. For example, the European Union FP7 Program supports the project "*Emergent Connectors for Eternal Software Intensive Networked Systems*", it studies the on-the-fly and emergent collaboration of heterogeneous software systems running in heterogeneous network environment. Both the NSF and PCAST (President Council of Advisors on Science and Technology) in USA suggest that the Cyber-Physical Systems[41] might bring revolutionary changes to software technologies for orchestrating computing process and physical process.

New "networked" operating systems also emerged. For example, the TinyOS[42] is developed for Wireless Sensor Networks, and Android and Apple iOS are developed for smart phones and tablets. To continuously meet the user preferences, new features can be on-demand loaded and customized. The resource management is another important issue of networked operating systems. Spurred by Cloud Computing, more and more resources (including hardware infrastructures, operating platforms, databases and applications) are concentrated in data centers. With virtualization technologies, these resources can be virtualized on demand to satisfy various applications.

Some new software development methodologies also emerged. For example, human factors have more significant impacts on the software development process, e.g., the user-generated contents and applications, social-networking, and collective intelligence from open source communities[43].

High-confidence also covers wide research interests. For example, the Trusted Computing Base, Trusted Computing, Open Trusted Computing, and High-Confidence Software, all support the "software confidence" from different perspectives and at different levels.

The Internetware paradigm has to accommodate itself to the new trends. Some research efforts are now in progress. For example, for the Internetware software model, the autonomous component model will be extended to support the Internet-of-Things environment. For the Internetware operating platform, current Internetware middleware will leverage resource virtualization and dynamic scheduling. New middleware is to be developed, such as the lightweight Web browser-based middleware running on different devices (PC, smart phone, and tablet like iPad). For the Internetware engineering approach, various quality analysis mechanisms are integrated to take comprehensive insights of quality. Also, end-user programming is employed to develop applications by users themselves.

5 Conclusion

This paper introduces the Internetware paradigm for Internet computing, and summarizes the research and practice of Internetware. The main achievements provide an architecture centric technical framework for Internetware paradigm, including the Internetware software model, the Internetware middleware, and the Internetware engineering approach.

The decade long research and practice on Internetware reveal the *WHAT-IS-TO-BE*, the *HOW-TO-DEVELOP*, and the *HOW-TO-RUN* issues of software in the Internet computing environment. As the Internet plays the central role in the complex network environment (the cyberspace), the *HOW-WELL* will become the key issue of Internetware before putting such a new software paradigm into new Internet-based applications. We believe that there are more potential research issues in software technology in the future.

References

[1] Mei H. Internetware: Challenges and future direction of software paradigm for Internet as a computer. In *Proc. the 34th IEEE Annual Computer Software and Applications Conference* (*COMPSAC*), Seoul, Korea, Jul. 19-23, 2010, pp.14-16.

[2] Yang F Q, Lü J, Mei H *et al.* Some discussion on the development of software technology. *Acta Elect Sin.*, 2003, 26(9): 1104-1115. (in Chinese)

[3] Lü J, Ma X X, Tao X P *et al.* Research and progress of Internetware. *Sci. China Ser. F*: *Info. Sci.* 2006, 36(10): 1037-1080. (in Chinese)

[4] Wang H M *et al.* Trustworthiness of Internet-based software. *Sci. China Ser. F*: *Info. Sci.*, 2006, 49(6): 759-773.

[5] Yang F Q, Lü J, Mei H. Technical framework for Internetware: An architecture centric approach. *Science in China Series F*: *Info. Sci.*, 2008, 51(6): 610-622.

[6] Jiao W P, Mei H. Automated adaptations to dynamic software architectures by using autonomous agents. *Eng. Appl. Art. Intell.*, 2004,17(7): 749-770.

[7] Mei H, Huang G *et al.* An architecture centric engineering approach to Internetware. *Sci. China Ser. F*: *Info. Sci.*, 2006, 49(6): 702-730.

[8] Jiao W P, Mei H. Supporting high interoperability of components by adopting an agent-based approach. *Software Qual. J.*, 2007, 15(3): 283-307.

[9] Jiao W P. Multiagent cooperation via reasoning about the behavior of others. *Comp. Intel.*, 2010, 26(1): 57-83.

[10] Mei H, Huang G *et al.* A software architecture centric self-adaptation approach for Internetware. *Sci. China Ser. F*: *Info. Sci.*, 2008, 51(6): 722-742.

[11] Lü J, Tao X P, Ma X X *et al.* Research on agent-based model for Internetware. *Sci. China Ser. F*: *Info. Sci.*, 2005, 35(12): 1233-1253. (in Chinese)

[12] Lü J, Ma X X, Tao X P *et al.* On environment-driven software model for Internetware. *Sci. China Ser. F*: *Info. Sci.*, 2008, 51(6): 683-721.

[13] Ma X X, Cheung S C, Cao C, Xu F, Lü J. Towards a Dependable Software Paradigm for Service-Oriented Computing. Dong J *et al.* (eds.), High Assurance Services Computing, Springer, DOI 10.1007/978-0-387-87658-0_9, pp.163-192.

[14] Mei H, Liu X Z. Software technologies for Internet computing: Current status and future outlook. *Chinese Sci. Bull.*, 2010, 55(31): 3510-3516.

[15] Issarny V, Caporuscio M, Georgantas N. A perspective on the future of middleware-based software engineering. In *Proc. Future of Software Engineering*, Minneapolis USA, May 23-25, 2007, pp.244-258.

[16] Huang G, Wang Q X, Cao D G *et al.* PKUAS: A domain-oriented component operating platform. *Acta. Elect. Sin.*, 2002, 30(12Z): 39-43. (in Chinese)

[17] Huang T, Chen J N, Wei J *et al.* OnceAS/Q: A QoS-enabled web application server. *J. Software*, 2004, 15(12): 1787-1799. (in Chinese)

[18] Huang T, Ding X N, Wei J. An application-semantics-based relaxed transaction model for Internetware. *Sci. China Ser. F*: *Info. Sci.*, 2006, 49(6): 774-791.

[19] Wang Q X, Shen J R, Wang X P *et al.* A component-based approach to online software evolution. *J. Software Main Evol.: Res. Prac.*, 2006, 18(3): 181-205.

[20] Shen J R, Sun X, Huang G *et al.* Towards a unified formal model for supporting mechanisms of dynamic component update. In *Proc. The Fifth Joint Meeting of the European Software Engineering Conference and ACM SIGSOFT Symposium on the Foundations of Software Engineering* (*ESEC-FSE 2005*), Lisbon, Portugal, Sept. 5-9, 2005, pp.80-89.

[21] Li J G, Chen X P, Huang G, Mei H, Franck C. Selecting fault tolerant styles for third-party components with model checking support. In *Proc. the 12th International Symp. Component-Based Software Engineering* (*CBSE*), East Stroudburg University, USA, Jun. 14-26, 2009, pp.69-86.

[22] Huang G, Mei H, Yang F Q. Runtime software architecture based on reflective middleware. *Sci. China Ser. F*: *Info. Sci.*, 2004, 47(5): 555-576.

[23] Huang G, Song H, Mei H. SM@RT: Applying architecture-based runtime management into Internetware systems. *Int. J. Softw. and Infor.*, 2009, 3(4): 439-464.

[24] Huang G, Liu X Z, Mei H. An online approach to feature interaction problems in middleware based systems. *Sci. China Ser. F*: *Info. Sci.*, 2008, 51(3): 225-239.

[25] Huang G, Liu T C, Mei H *et al.* Towards autonomic computing middleware via reflection. In *Proc. the 28th Annual International Computer Software and Applications Conference* (*COMPSAC*), Hong Kong, China, Sept. 28-30,2004, pp.122-127.

[26] Cao J N, Feng X Y, Lü J *et al.* Reliable message delivery for mobile agents: Push or pull? *IEEE Trans. Syst. Man. Cyber., Part A: Systems and Humans*, 2004, 34(5): 577-587.

[27] Jin Z, Lu R Q. Automated requirements modeling and analysis: An ontology-based approach. *Sci. China Ser. E*, 2003, 33(4): 297-312. (in Chinese)

[28] Hou L S, Jin Z, Wu B D. Modeling and verifying web services driven by requirements: An ontology-based approach. *Sci. China Ser. F*: *Info. Sci.*, 2006, 49(6): 792-820.

[29] Mei H, Zhang W, Zhao H Y. A metamodel for modeling system features and their refinement, constraint and interaction relationships. *Software Syst. Mod.*, 2006, 5(2): 172-186.

[30] Zhang W, Mei H. A feature-oriented domain model and its modeling process. *J. Software*, 2003, 14(8): 1345-1356. (in Chinese)

[31] Cui X F, Sun Y C, Mei H. Towards automated solution synthesis and rationale capture in decision-centric architecture design. In *Proc. Working IEEE/IFIP Conference on Software Architecture*, Vancouver, Canada, Feb. 18-22, 2008, pp.221-230.

[32] Zhao W, Zhang L, Liu Y *et al.* SNIAFL: Towards a static non-interactive approach to feature location. *ACM Trans. Software Engin. Meth.*, 2006, 15(2): 195-226.

[33] Pan Y, Wang L, Zhang L *et al.* Relevancy based semantic interoperation of reuse repositories. In *Proc. the 12th ACM SIGSOFT Symposium on Foundations of Software Engineering* (*FSE2004*), Delhi, India, Feb. 5-7, 2004, pp.211-220.

[34] Zhong H, Zhang L, Xie T, Mei H. Inferring resource specifications from natural language API documentation. In *Proc. the 24th IEEE/ACM International Conf. Automated Software Engineering* (*ASE 2009*), Auckland, New Zealand, Nov. 16-20, 2009, pp.307-318.

[35] Liu X Z, Huang G, Mei H. Discovering homogeneous web services community in the user-centric web environment. *IEEE Trans. Serv. Comput.*, 2009, 2(2): 167-181.

[36] Mei H, Chang J C, Yang F Q. Software component composition based on ADL and middleware. *Sci. China Ser. F*: *Info. Sci.*, 2001, 44(2): 136-151.

[37] Cao J N, Ma X X, Chan A T S, Lü J. Architecting and implementing distributed web applications using the graph-oriented approach. *Software-Practice & Experience*, 2003, 33(9): 799-820.

[38] Liu X Z, Huang G, Mei H. A community-centric approach to automated services composition. *Sci. China Ser. F*: *Info. Sci.*, 2010, 53(1): 50-63.

[39] Lu R Q. From hardware to software to knowware: IT's third liberation? *IEEE Intell. Syst.*, 2005, 20(2): 82-85.

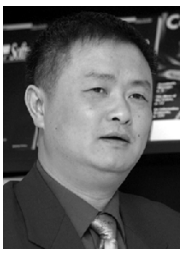
[40] Lu R Q, Jin Z. From knowledge based software engineering to knowware based software engineering. *Sci. China Ser. F*: *Info. Sci.*, 2008, 51(6): 638-660.

[41] Edward A Lee. Cyber-physical systems: Are computing foundations adequate? *NSF Workshop on Cyber-Physical Systems: Research Motivation, Techniques and Roadmap*, Position Paper, Oct. 16-17, 2006, Austin, USA.

[42] Polastre J, Hui J, Levis P, Zhao J, Culler D E, Shenker S, Stoica I. A unifying link abstraction for wireless sensor networks.

In *Proc. ACM SenSys 2005*, San Diego, USA, Nov. 2-4, 2005, pp.76-89.

[43] Herbsleb J D. Global software engineering: The future of socio-technical coordination. In *Proc. Future of Software Engineering* (*FOSE 2007*), Minneapolis, USA, May 23-25, 2007, pp.188-199.



Hong Mei is a full professor of School of Electronics Engineering and Computer Science, Peking University, China. He received the Ph.D. degree in computer science from Shanghai Jiao Tong University in 1992. His current research interests are in the area of software engineering, software reuse and software component technology, and distributed object technologies. He is a member of the Expert Committee for Information Technology of National High-Tech Research and Development 863 Program of China, a chief scientist of National Basic Research 973 Program of China, and the director of Technical Committee on System Software (TCSS) of China Computer Federation (CCF). He is also a fellow of CCF.

Xuan-Zhe Liu is an assistant professor in the School of Electronics Engineering and Computer Science, Peking University, China. He received his Ph.D. degree from Peking University in 2009. His research interests are in the area of software engineering, services computing, and social computing. He is a CCF member.