

# Internetware: A Software Paradigm for Internet Computing

Hong Mei, Gang Huang, and Tao Xie, *Peking University, China*

**To meet the needs of computing in the Internet environment, the Internetware software paradigm provides a set of technologies that support the development of applications with characteristics that are autonomous, cooperative, situational, evolvable, emergent, and trustworthy.**

**D**ue to its open, dynamic, constantly changing nature, the Internet computing environment exhibits characteristics that call for new software development technologies, reflecting the pattern of ever-evolving paradigms throughout computing history.

A software paradigm (also called a programming paradigm) describes a software model and its construction from the perspective of software engineers or programmers.<sup>1</sup> A software model specifies the forms, structures, and behaviors of software entities as well as their collaborations.

As Figure 1 shows, software collaborations typically occur via predefined or dynamic interactions among software entities. In a structured paradigm, these entities are procedures, and their interactions occur through predefined procedural calls. In an object-oriented paradigm, object interactions occur through message passing. In a component-based paradigm, component interactions occur through connectors.

In a service-oriented paradigm, different services can interact with each other to support the same requirement.

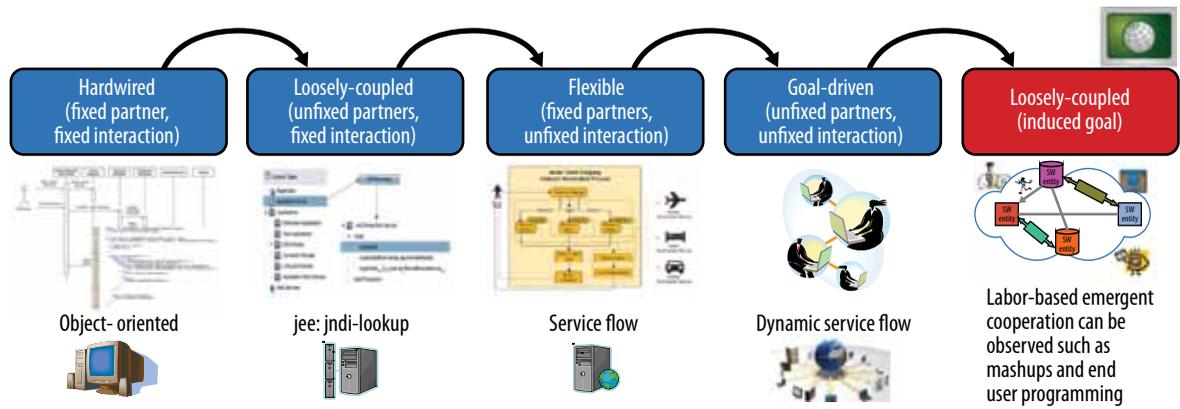
For example, the OASIS Web Services Business Process Execution Language (WS-BPEL) supports selecting one of several functionally equivalent candidate services for composition; it also supports selecting one of several interaction styles from those services.

However, in the Internet computing environment, requirements are often unclear or undetermined before collaboration occurs. For example, handling a swine flu epidemic requires cooperation among numerous organizations and individual software and services, such as airlines, hotels, hospitals, and mobile phone service providers. Similar scenarios arise in both emergency situations (such as earthquakes, typhoons, or snow disasters) and during major international or national events (such as the Olympic Games or the National Day celebration).

The requirements involved in handling these types of situations call for new technologies that can support on-demand collaborations among software entities. Chinese researchers have proposed Internetware,<sup>1-5</sup> a software paradigm that provides a set of technologies for developing applications to meet computing requirements in the Internet environment. The “China’s National Research and Development Framework” sidebar offers more information on the current status of projects related to the development of Internetware.

## INTERNET COMPUTING REQUIREMENTS

Software on the Internet differs from traditional software in terms of its form, structure, and behavior. Consequently, software applications (including software entities and their interactions) for Internet computing



**Figure 1. Paradigms enabling software collaborations.** Such collaborations typically occur via predefined or dynamic interactions among software entities.

should be autonomous, cooperative, situational, evolvable, emergent, and trustworthy.<sup>1</sup>

### Autonomous

Software entities are usually distributed, relatively self-contained, and independent. They usually perform according to provider-defined composition or deployment strategies and strive continuously to satisfy provider requirements. A software entity can adapt itself when necessary by sensing and collecting information on environment changes. For example, in the swine flu epidemic scenario, the Centers for Disease Control (CDC), airports, hotels, and hospitals are all autonomous entities and would need to participate in on-demand collaboration with each other.

### Cooperative

A set of software entities can collaborate for the purpose of business or management.<sup>6</sup> Often, the collaboration is dynamic rather than static to adapt to on-demand user requirements and environments. The collaboration mechanisms between software entities can be of various types and can change if necessary.

### Situational

Software applications can be aware of runtime contexts and scenarios, including the underlying devices, operating platforms, networking conditions, application contexts,<sup>7</sup> or changes in other dependent applications. Hence, both software entities and their operating platforms need to be capable of exposing their runtime states and behaviors in some way.

### Evolvable

The structures and behaviors of software applications might change dynamically. Internet software applications usually consist of autonomous entities that provide continuous online services 24/7 for numerous users. Hence,

software applications cannot be shut down during evolution. They must perform what amounts to an online evolution to accommodate new user requirements and environments. Possible evolutions can include the addition or removal of software entities; just-in-time, on-the-fly changes in functionalities; and changes in interaction styles and topologies among entities.

### Emergent

Software applications can exhibit random behaviors or undesired effects at runtime. Such behaviors might iteratively result in more changes in software-application structures and behaviors to accommodate emergent requirements.

### Trustworthy

Software applications should promise comprehensive tradeoffs among various quality attributes. As software applications serve many online users, those applications' trustworthiness should cover a wide spectrum, including reliability, security, performance, and user experience. Quality assurance can stretch to include autonomous entities, interaction styles, network environments, usage patterns, malicious attacks, and software evolution.

## INTERNET COMPUTING PARADIGMS

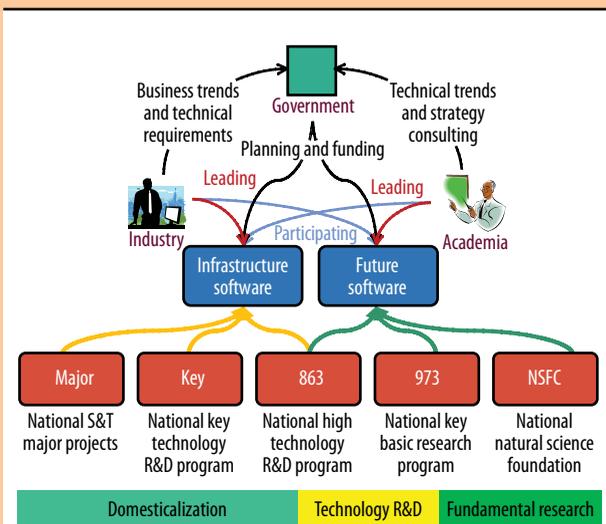
A software paradigm usually includes four main aspects: what to be, how to do, how to run, and how well.<sup>1</sup>

The what-to-be aspect refers to what is being constructed and executed—say, a software or programming model. How to do refers to how to develop the required software applications, including programming languages, engineering approaches, and supporting tools. How to run refers to the tools for running software applications, including entities such as runtime systems, operating systems, and middleware platforms. How well refers to how well the constructed and executed software applications can perform—their promised qualities,

## CHINA'S NATIONAL RESEARCH AND DEVELOPMENT FRAMEWORK

China's National Long-Term Science and Technology Development Plan (2006-2020) documents the following national research and development (R&D) strategies: independent innovation, infrastructure technology, future technology, and leap-forward development. Based on these four strategies, the Chinese government has initiated the development of a national R&D framework. As Figure A shows, the government uses this framework to plan the R&D roadmap and attract funding investment. Industry proposes business trends and technical requirements, while academia proposes technical trends and provides strategic consulting.

The R&D framework includes five major national programs. To satisfy current domestic market needs, the National Science and Technology Major Projects focus on products or systems such as airplanes and infrastructure products. The National Key Technology R&D Program focuses on technologies that are critical to the country's economy. The NSFC supports fundamental research. In addition to fundamental research, the National Basic Research Program (or 973) supports technology R&D that makes research results practical. The National High Technology R&D Program (or 863) focuses on advanced technology R&D.



**Figure A. China's national research and development (R&D) framework and its software technologies components.**

In the national R&D framework, the components related to software technologies focus on two categories: domestic production of infrastructure software and innovation in future Internet-related software technologies.

Based on worldwide technical trends and increasing Chinese informational demands, five types of infrastructure software are the major targets in domestic technology development: operating systems, database management systems, embedded systems software, middleware, and office suites. Infrastructure software technologies have been mature for many years, but domestic infrastructure software has a long way to go to meet the needs of the domestic market. For example, operating systems occupied no more than 5 percent of the domestic market share in 2005, and most of those purchases were through government procurement.

Because of its rapid pervasiveness and wide application, the Internet is becoming open, global, ubiquitous, and smarter. This trend calls for disruptive innovations in most aspects of the development of software technologies.

In the past decade, the Chinese National Basic Research Program (known as 973) has funded two consecutively executed projects to support the development of Internetware: 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 (2009-2013).

Approximately 80 researchers from Chinese universities and institutes have participated in these projects, including participants from Peking University, Nanjing University, Tsinghua University, the 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. The annual Asia-Pacific Symposium on Internetware ([www.internetware.org](http://www.internetware.org)), which debuted in 2009 in cooperation with ACM SIGSOFT, attracts authors and attendees from China, the US, Europe, Australia, Japan, and Korea.

In the past five years, the National Natural Science Foundation of China (NSFC) has sponsored deep research into Internetware topics such as trustworthiness and ontology development. The National High Technology Research and Development Program (known as 863) has sponsored research into some technical challenges in practice, such as self-adaption and self-organization. Researchers are introducing prototype Internetware operating platforms and development tools in commercial products with support from the National Science and Technology Major Projects.

correctness, performance, reliability, and anticipated user experiences.

A software paradigm's main objective is to better leverage the underlying runtime environment while offering a computing model that is sufficiently expressive and natural to characterize the application domain. Put another way, a software paradigm evolves with its underlying runtime environment and its target application domains. For a software paradigm, the what-to-be and how-to-do/how-to-run aspects are typically the first set of concerns that developers address, with how well becoming a concern when they broadly apply the software paradigm in practice.

Every shift in a software paradigm typically brings significant challenges as well as tremendous opportunities—think of the shift from the structured to the object-oriented paradigm, followed by the shift to the component-based/service-oriented paradigm. After more than a decade of development and growth, the expanded Internet computing environment and application domains call for a new shift in software paradigms.

In particular, an Internet computing software paradigm must satisfy the following requirements, some of which are general to any software paradigm and some of which are specific to this evolving environment.<sup>5</sup>

## Software model (what to be)

The software model should specify the forms, structures, and behaviors of software entities as well as their collaborations. These specifications determine the principles and features of the corresponding software technologies, including programming languages, development approaches, and runtime mechanisms. Developers can build basic Internet software entities using current technologies, such as the object-oriented and service-computing paradigms, but they should provide new capabilities to enable on-demand collaborations among entities along with context- and situation-aware capabilities.

## Software operating platform (how to run)

The operating platform should provide a runtime space for software entities and their collaborations. To ease the migration to applications for Internet computing, the operating platform should conveniently equip legacy software with new characteristics that satisfy the requirements in this environment. In addition, the operating platform should manage software applications and the platform itself intelligently and automatically.

## Engineering approach (how to do)

The engineering approach should systematically control the entire life cycle of developing software for Internet computing, including requirements specification, design, implementation, deployment, and maintenance.

## Quality assessment (how well)

Software applications on the Internet usually serve a large number of online users simultaneously.<sup>8</sup> Both quantitative and qualitative assessment methods should be developed for various quality attributes such as performance, reliability, and usability, and to enable comprehensive tradeoffs among these attributes.

## THE INTERNETWARE PARADIGM

The Internetware software model consists of a set of autonomous software entities distributed over the Internet, together with a set of connectors to enable collaborations among these entities in various ways. Internetware software entities can sense dynamic changes in the running environment and continuously adapt to these changes through structural and behavioral evolutions.

From the micro perspective, Internetware software entities can collaborate with each other on demand; from the macro perspective, these entities can self-organize to form an application or community of interest. Consequently, the development of a software application with Internetware can be viewed as a continuous and iterative composition of various “disordered” resources into “ordered” software applications. Thus, software development with Internetware is a spiral process built bottom-up and inside-out.

In particular, the Internetware software paradigm covers three main aspects.<sup>9</sup>

## Software model (what to be)

The Internetware software model deals with entities, collaborations, and environments, as well as their relationships. An Internetware software entity has basic business functionality interfaces to enable collaboration—it can expose its own states and behaviors, and it can also monitor and capture environment information. Entity collaborations governed by a software architecture can be globally planned and adapted.

## Software operating platform (how to run)

In Internetware middleware, software entity containers provide advanced capabilities and services. For example, flexible connectors can mediate different protocols between entities; the dynamic binding of policies satisfies the specified constraints.<sup>10</sup> A runtime software architecture (RSA)



**Internetware software entities can sense dynamic changes in the running environment and continuously adapt to these changes through structural and behavioral evolutions.**

governs on-demand collaborations. Leveraging autonomic computing for management, Internetware middleware supports the self-organization and self-adaptation of software applications, and supports the delivery of high quality of service (such as reliability and performance) at runtime. In particular, the middleware is open and extensible, so it can load or customize new capabilities and services on-demand.

## Engineering approach (how to do)

The Internetware engineering approach follows the core principle of “software architecture for the whole life cycle.” Essentially, the software architecture serves as a blueprint and controls every stage of software development. To support the online self-organization and self-adaptation of Internetware software applications, the architecture implements and governs software entities and their on-demand collaborations. To better control the development process, the architecture organizes heterogeneous distributed resources for a specific domain according to domain modeling techniques.

## INTERNETWARE TECHNOLOGIES AND INFRASTRUCTURES

As Figure 2 shows, researchers in the Chinese software community have developed various key technologies and infrastructures to support Internetware.

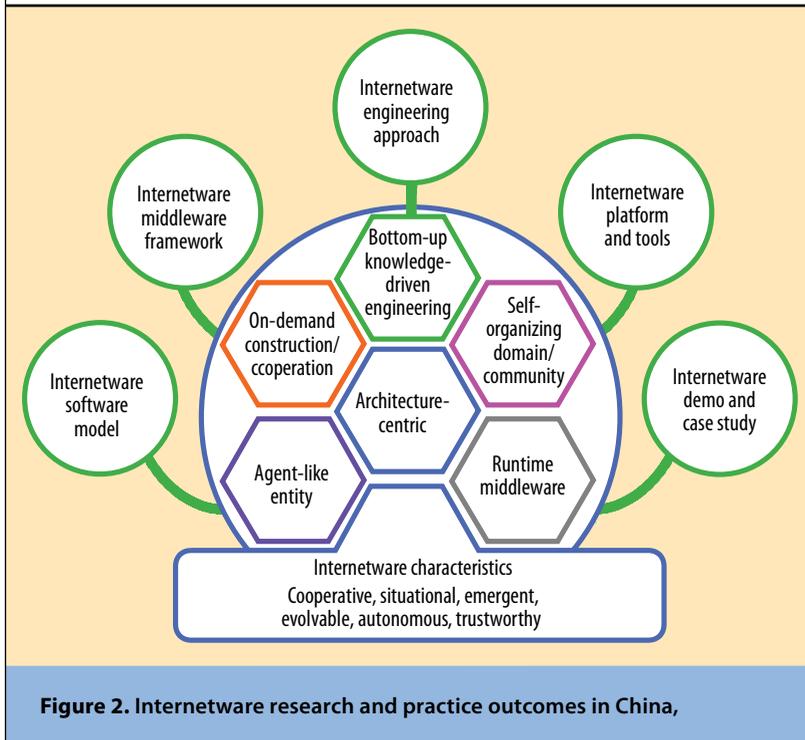


Figure 2. Internetwork research and practice outcomes in China,

### Software model

One recently proposed autonomous component model is built on current software technologies, including EJB or Web service components.<sup>11</sup> The entities include the metamodel component and reflective interfaces for adapting states and behaviors. The entities can further leverage the environment metamodel and reflective interfaces to capture contextual information.<sup>12</sup> Entities with a rule engine can reason about proper actions based on context and defined rules to perform adaptations intelligently and automatically.

### Operating platform

The JavaEE-compliant Peking University Application Server (PKUAS)<sup>13</sup> is an architecture-based reflective component operating platform that provides the Internetware software container. PKUAS provides an open framework to facilitate interaction by heterogeneous components with different interoperability protocols, such as RMI, SOAP, and HTTP. It also allows flexible binding of various elements such as security policies and database connections for rule-based component self-adaptation.

Furthermore, PKUAS allows the dynamic addition or removal of components as well as upgrading or downgrading connectors, and supports online evolution of Internetware applications. To support on-demand collaboration, Internetware middleware employs an RSA and reflection mechanisms on its own application server. Based on the RSA, SM@RT (software model at runtime), an architecture-based, model-driven, self-adaption technique, supports

autonomic management of the Internetware middleware and the software applications operating on it.

### Engineering approach

Internetware's engineering approach leverages the architecture-based component composition (ABC) methodology.<sup>11</sup> ABC composes reusable components under a single blueprint software architecture and shortens the gap between high-level design and low-level implementation. ABC's adaptation capability is crucial to satisfying Internet computing's software requirements. We use a software architecture to locate the entities to be adapted, then the software architecture records what should happen at runtime to adapt those entities. Finally, the software architecture executes the designed adaptation without stopping the running system. We can also integrate reasoning rules into the software architecture to enable dynamic adaptation with a rule engine.

**R**esearchers are investigating several issues related to Internetware.

The Internet is playing an increasingly essential role in connecting the cyber, physical, and social worlds. Developers must extend Internetware to cope with the complex, software-intensive cyberspace resulting from the connections and collaborations between telecom, mobile, sensor, and other ad hoc networks. Computing devices, human society, and physical objects will soon be seamlessly integrated together, and software systems will orchestrate information, processes, decisions, and interactions in this Internet environment. Internetware must be ready to support the software systems that will only continue to grow in scale and complexity in this space.

Researchers also must extend Internetware to accommodate the shifted focus on software quality. In the evolving cyberspace, software systems will directly serve millions or even billions of users accessing various online services. The diversity of network environments, devices, and user preferences make quality assurance much more challenging and complex. Internetware software applications and their operating platform should offer good enough, cost-effective (rather than best-effort) assurance.

In addition, the Internetware software paradigm must accommodate emerging application domains and provide support for the Internet-of-Things environment. The

Internetware middleware must leverage resource virtualization and dynamic scheduling. A lightweight version of the middleware must be based on Web browsers so that it can run on both PCs and various mobile devices, such as smartphones and tablets. Ultimately, Internetware must support end user programming so that the users themselves can engage in the engineering process. **C**

## Acknowledgments

This work is sponsored by the National Basic Research Program of China under grant no. 2009CB320700; the National Natural Science Foundation of China under grant no. 61121063, 91118004, 60933003; the European Commission Seventh Framework Programme under grant no. 231167; and the IBM University Joint Study.

## References

1. H. Mei, "Internetware: Challenges and Future Direction of Software Paradigm for Internet as a Computer," *Proc. 34th Ann. Conf. Computers, Software, and Applications (CompSAC 10)*, 2010, pp. 14-16.
2. F. Yang, J. Lü, and H. Mei, "Some Discussion on the Development of Software Technology," *Acta Electronica Sinica* (in Chinese), vol. 26, no. 9, 2003, pp. 1104-1115.
3. J. Lü, X. Ma, and X-P. Tao, "Research and Progress of Internetware," *Science in China*, series F (in Chinese), vol. 36, no. 10, 2006, pp. 1037-1080.
4. H. Mei and X. Liu, "Internetware: An Emerging Software Paradigm for Internet Computing," *J. Computer Science and Technology*, vol. 26, no. 4, 2011, pp. 588-599.
5. J. Lü et al., "Internetware: A Shift of Software Paradigm," *Proc. 1st Asia-Pacific Symp. Internetware*, ACM, 2009; doi:10.1145/1640206.1640213.
6. T. Liu, Ying Li, and X. Li, "A Collaborative Management as a Service Framework for Managing Distributed Systems," *Proc. 1st Asia-Pacific Symp. Internetware*, ACM, 2009; doi:10.1145/1640206.1640218.
7. C. Ye et al., "A Study on the Replaceability of Context-Aware Middleware," *Proc. 1st Asia-Pacific Symp. Internetware*, ACM, 2009; doi:10.1145/1640206.1640210.
8. H. Wang et al., "Trustworthiness of Internet-Based Software," *Science in China*, series F, vol. 49, no. 6, 2006, pp. 759-773.
9. F. Yang, J. Lü, and H. Mei, "Technical Framework for Internetware: An Architecture Centric Approach," *Science in China*, series F, vol. 51, no. 6, 2008, pp. 610-622.
10. C. Ye et al., "Middleware Support for Internetware: A Service Perspective," *Proc. 2nd Asia-Pacific Symp. Internetware*, ACM, 2010; doi:10.1145/2020723.2020727.
11. H. Mei et al., "A Software Architecture-Centric Engineering Approach for Internetware," *Science in China*, series F, vol. 49, no. 6, 2006, pp. 702-730.
12. J. Lü et al., "An Environment-Driven Software Model for Internetware," *Science in China*, series F, vol. 51, no. 6, 2008, pp. 683-721.
13. H. Mei et al., "A Software Architecture-Centric Self-Adaptation Approach for Internetware," *Science in China*, series F, vol. 51, no. 6, 2008, pp. 722-742.

**Hong Mei** is a professor in the School of Electronics Engineering and Computer Science, Peking University. His research interests include software engineering, software reuse, distributed object technology and middleware, and programming languages. Mei received a PhD in computer science from Shanghai Jiaotong University. He is a senior member of IEEE. Contact him at [meih@pku.edu.cn](mailto:meih@pku.edu.cn).

**Gang Huang** is a professor in the School of Electronics Engineering and Computer Science, Peking University. His research interests include software engineering, particularly software architecture and middleware. Huang received a PhD in computer science from Peking University. He is a member of IEEE. Contact him at [hg@pku.edu.cn](mailto:hg@pku.edu.cn).

**Tao Xie** is a visiting professor in the School of Electronics Engineering and Computer Science, Peking University, and an associate professor in the Department of Computer Science at North Carolina State University. His research interests include software engineering, particularly software testing, program analysis, and software analytics. Xie received a PhD in computer science from the University of Washington, Seattle. He is a member of IEEE and a senior member of ACM. Contact him at [xie@csc.ncsu.edu](mailto:xie@csc.ncsu.edu).



Selected CS articles and columns are available for free at <http://ComputingNow.computer.org>.