SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •
Special Focus on Software Engineering for Internet Computing

August 2016, Vol. 59 080103:1–080103:13
doi: 10.1007/s11432-016-5596-7

Flourishing creativity in software development via Internetware paradigm

Lin LIU[1]*, Letong FENG[1], Yanqing LI[1], Delin JING[2] & Hongji YANG[2]

[1]*School of Software, Tsinghua University, Beijing* 100084*, China;*
[2]*Centre for Creative Computing, Bath Spa University, Corsham* SN13 0BZ*, England, UK*

Received April 25, 2016; accepted June 7, 2016; published online July 18, 2016

Abstract While culture being the software controlling human mind, computer software development becomes one of the most creative activities that human undertake since the civilisation began. The only limitation in software creation is human imagination, and that limit is often self-imposed. The "Internetware", referring to a software paradigm, aims to satisfy the need of human kind using Internet as an integrated development and execution platform. Such software systems are composed of entities distributed through the Internetwork, allowing connections that would be impossible or difficult to make otherwise. One of the tasks for the Internetware is to accommodate creativity, to understand the general settings of creative design process and to develop programs that can enhance creativity without necessarily being creative themselves. Therefore, it can be summarized that a development environment needs to be built to best support software creation process of six steps including searching, ideating, specifying, coding, testing and evolving. An E-Health application eco-system is used to illustrate the proposed development process model.

Keywords requirements, information system, Internetware, creativity, evolution

Citation Liu L, Feng L T, Li Y Q, et al. Flourishing creativity in software development via Internetware paradigm. Sci China Inf Sci, 2016, 59(8): 080103, doi: 10.1007/s11432-016-5596-7

1 Introduction

Software creativity occurs when designers add new solutions that are previously unknown to the space of design alternatives. The Internetware paradigm [1–4] provides a compositional creativity platform for software designers, where they have the freedom and facility to access an extensive repository of design artifacts, to explore alternative possible designs, and to choose a preferred design in response to emerging requirements. The traditional model-based transformation tools mapping requirements models to system implementations across levels of abstractions are substituted by simple end-user programming tools, which allow the construction and deployment of new applications based on existing frameworks with minimum program effort. End user testing makes online data collection and fast turnaround a matter of fact. Internet based software development environment has made many inherent sustainable software eco-systems, which makes building software systems online faster and easier with contemporary automated tools, reusable code assets and community support.

* Corresponding author (email: linliu@tsinghua.edu.cn)

It has been observed that computing, particularly, its programming part is creative in terms being conducted in a dynamic, and often collaborative, process. While many think Internetware is an evolutionary product of the traditional networked software systems, we argue that it has brought revolutionised changes to software development life cycle and software development behaviour of engineers. Among many characteristics, such as autonomy, evolutionary, collaborative, polymorphic, quickly spreading, we consider collaboration the most valuable and preferable as today's software system development requires incremental creativity, which means building upon each other's intellectual property, collaborating to achieve a common goal, and taking advantage of each other's know-hows than wasting time and effort reinventing the wheels. There are coordination mechanisms between them. The inherent *evolutionary* property of Internetware requires effective way to manage the diversity of user requirements and fast absorbance of available technologies; in other words, users need an accessible supporting platform and evaluation environment for building and sharing interesting software products.

Conventional software lifecycle is explicitly divided into design time and run-time stage. Today's web-based systems evolve without going offline to cope with constant arriving application requests, and the boundary between design-time and run-time are almost unnoticeable to users. This paper introduces a creativity enhanced software lifecycle model in the Internetware setting. Section 2 describes the major steps of software development cycle for Internetware systems, in comparison with traditional software lifecycle. Section 3 introduces the adaptation process and major algorithms. Section 4 uses a case study to illustrate and evaluate the proposed approach and the tools we have built to support the proposed process. Section 5 discusses related work and Section 6 summarises the contributions of the paper.

2 Software development in the Internetware

We propose that an Internetware environment to be built should: (1) support divergent and convergent thinking, i.e., possible solutions are proposed first and then the preferred one is chosen, (2) support reflexivity (reflection, planning and action adaptation) knowing how well one is doing. It is proposed in this paper that the following steps are forming the sequence or the life cycle of software development in the Internetware Paradigm:

- Searching—for similar exiting approaches
- Ideating—combining individual or mass creativities to form ideas
- Specifying—user requirements
- Coding—maximising reuse of existing (open source) codes
- Operation—ready for evolving.

To demonstrate what features an Internetware platform shall accommodate, a prototype of such has been designed and experimented (Figure 1). With the support of an Internetware platform, users get access to required information and entities easily and timely. As new requirements and software assets emerge, problem-solution bindings are formed, updated and optimised continuously. The system is yet to be fully realised due to the diversity and constant changes of requirements, limited reliable repositories available, and the lacking of a supporting platform matching requirements with available entities. In order to achieve a common understanding to the requirements analysis for Internetware systems, we shall notice that there is a shift of paradigms from conventional design-time requirements, to run-time reaction according to the environmental changes, and on-demand requirements and capability match-making and composition. The fundamental goal is to explicitly point out what has to be defined prior to running, and what is changeable during run time, how to make the system self-adapt to run-time demands.

3 Searching: looking for similar problems or solutions

The first stage of software life cycle for Internetware paradigm is (re)searching. "Searching" seems to be part of the life nowadays, with the evidence that people are using search engines daily for any topic. And the word "searching" seems to be used interchangeably with "researching", which comprises creative

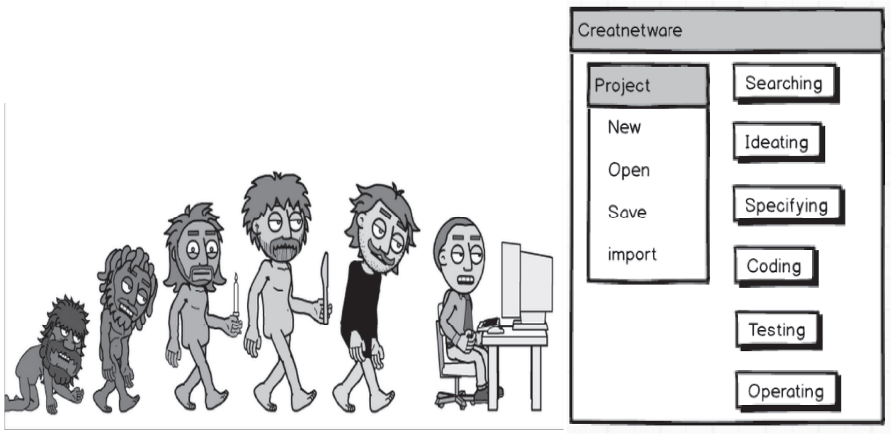


Figure 1 Six stages of Internetware on the Go.

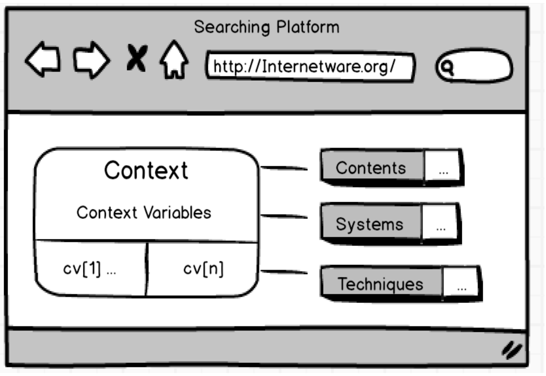


Figure 2 Search for contents, systems, techniques under given context.

work undertaken on a systematic basis in order to increase the stock of knowledge, including knowledge of man, culture and society, and the use of this stock of knowledge to devise new applications. Apparently, data, information and knowledge are rich and easy to reach. This implies that, on one hand, good ideas can be disseminated rapidly and on the other hand new ideas are much fewer and far in between. To start with, one needs to be certain any idea is new, which can be done naturally by (re)searching. Secondly, the knowledge through search can be used to enhance the original idea. The environment to be built ought to accommodate a thorough search, which is to establish or confirm facts, reaffirm the results of previous work, solve new or existing problems, support or develop new theories and methods and applications, and finally record search results (Figure 2).

• Context—under which a system is to be built and operate. It is assumed that their individual background, organisational roles, and the immediate operation context together determine the needs and preference of developers and users. In existing context-aware computing literature, context information is often pre-assumed to include a limited number of variables, e.g., time, location, and run-time status of the platform. So each search action is based on a given context description, comprised of propositions on context variables under observation.

• Contents—search for digital contents with desired features under given context. This includes learning of domain knowledge, collection of data in varied forms and usage purposes. For example, if an E-Health application for a dental office is to be built, the basic clinical document structures, fundamental medical terms, graphical oral 2D, 3D navigation models, and so on, have to be in place, which either come from the customer organisation, or retrieved from reliable web sources.

• Systems—no matter how innovative a system under conceiving is, there can be existing systems similar to it in terms of functions and operation scenarios. For a given software type, and a set of requirements, similar product can be found via general search engines or specific online service repositories, e.g., an example query might be online Customer Relationship Management (CRM) software. Possible steps include: search for the accessible web products, use CRM software evaluation tools based on the current business requirements of the user, and generate a report and/or a ranking amongst the available tools based on user defined criteria.

• Techniques—there are two types of innovations driving the development of the software industry, technological innovation and application innovation. The earlier one looks into the innovation of relevant engineering knowledge and software techniques, e.g., programming languages, software architectures, etc. The latter, on the other hand, covers the socio-techno side of engineering knowledge, which is a necessary condition for the success in the industry setting.

4 Ideation—software creation: by individual or by mass collaboration?

The second stage of software life cycle for Internetware is ideation, which is a creative process, following the search stage, of generating, developing, and communicating new ideas, where an idea is understood as a basic element of thought that can be either concrete or abstract. Ideation comprises all stages of a thought cycle, from innovation, to development, to actualisation.

(a) Domain Knowledge. Presently, people are taking advantage of the technology to communicate in various ways easily, along with which new ideas are generated. It is considered that the environment proposed should accommodate both individual and mass creativities. Creating ideas can be complicated, which may need a large amount of knowledge as support. In this paper, work is narrowed down to specific domains. Therefore, various knowledge is required from each relevant domain.

An approach [5] is proposed to generate ideas with the help of a domain ontology, in which the gap between knowledge collection and mental thought is bridged. A designed method can be employed to assist on extracting ontology as knowledge for specific domains [6]. Then, based on knowledge combination, ideas are formed supported by other techniques, which are required by applications. It is obvious that the volume of knowledge directly affects the result ideas. Ontology, as a formal representation method with reasoning potential, can be a way to increase the possibility and efficiency on generating new ideas.

Abstraction techniques are used to collect, extract and organise domain knowledge from structured or unstructured information. There are also some rules [5] helping on mapping abstracted information to ontology format. Based on the techniques and rules, a relatively complex process is needed to cover text extracting activities and ontology formatting activities, such as text filter, words cluster, vocabulary mapping, and so on.

Ideation is about generating new and applicable ideas. Thus, creativity is considered as an essential feature. Many studies indicated that creativity provides many advantages in companies [7]. As the approach [5] proposed, abstraction is the beginning of the idea creation to prepare necessary knowledge, whilst a proper extraction method is required as the kernel of the abstraction part.

Many techniques are designed to extract data, e.g., webpage annotation technique [8,9]. Obviously, it is not realistic to extract knowledge manually from a great amount of documents. Besides, it lacks developed tools to achieve the required extraction. Therefore, a designed abstraction method [5] is adopted, which provides simple rules to follow and automatic potential to extract knowledge as ontology from vast amount of informal structural information. Figure 14 shows an example of using domain ontology in a system.

It is believed that Ideation includes Idea generation and Idea Evolution. There are three kernel phases proposed for the approach of ideas creation including "Domain Ontology Extraction", "Idea Generation" and "Idea Evolution" [5] (Figure 3). The domain knowledge is prepared to serve the following phases.

(b) Idea Generation. Ideation Brainstorming is a guided, knowledge-based brainstorming approach that combines the advantages of Osborn's traditional problem-solving techniques. Ideation Brainstorming is generally applicable in any technological domain. As an ideation platform (Figure 4), Internetware is essentially a process of diversion thinking. Based on the information obtained from the previous step and the developed domain ontology, initial ideas are generated by participants of the platform as well as an automated inference engine, which deploys algorithms combining concepts with creativity features. Key

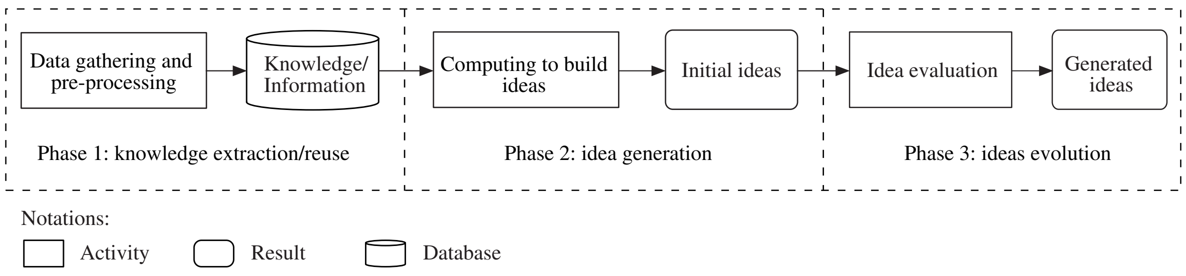


Figure 3 Ideas creation process [5].

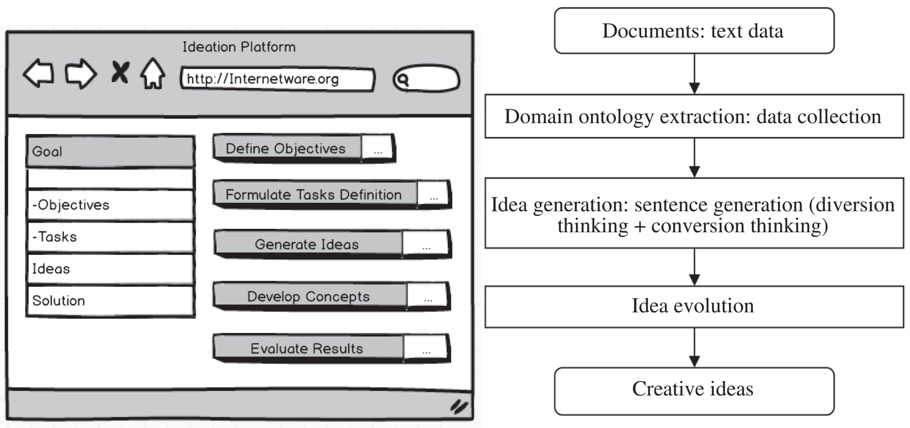


Figure 4 Ideation platform.

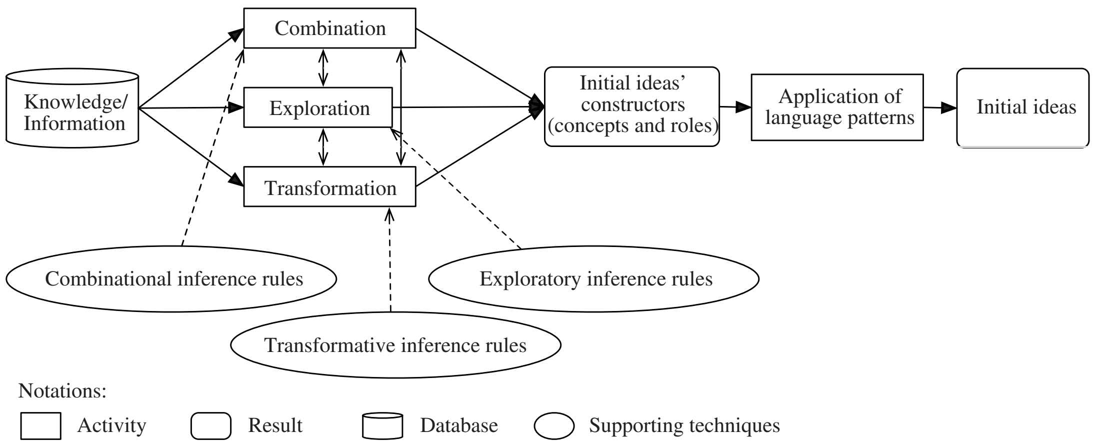

Knowledge/ Information
Combination
Exploration
Transformation
Initial ideas' constructors (concepts and roles)
Application of language patterns
Initial ideas
Combinational inference rules
Exploratory inference rules
Transformative inference rules
Notations:
Activity
Result
Database
Supporting techniques

Figure 5 Idea generation process [6].

factors here are concepts and relationships.

In particular, the process of idea generation is shown as Figure 5. Based on the domain knowledge gathered and processed via designed abstraction method, the system computes based on selected and designed methods to achieve diversion thinking and form new ideas. By employing Creative Computing, exploration, transformation and combination are adopted as kernel activities of its computing step to generate initial ideas [6]. Exploration activity looks for possibilities within a matured domain by searching relevant knowledge base. Combination activity is to gather and unite familiar knowledge via unfamiliar combinations. Transformation activity is about twisting and rejecting of constraints and assumptions. These activities support applications to compute domain knowledge and generate basis of new ideas as concepts and roles. Then, by applying certain language patterns, the ideas' constructors are formed as ideas that are understandable for users. Besides, it is capable to generate ideas through multi-activities

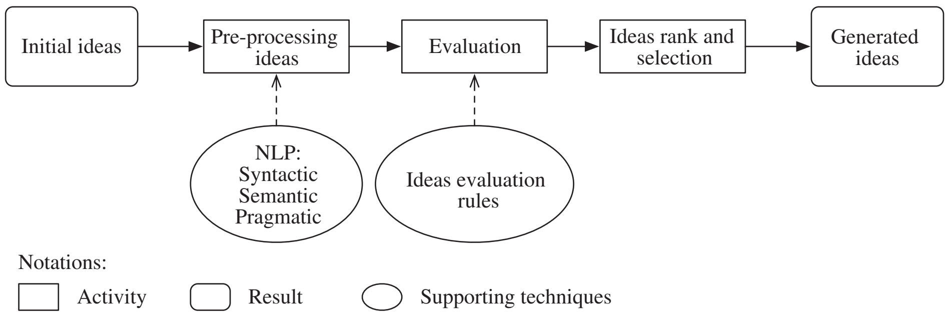


Figure 6 Idea evolution process [6].

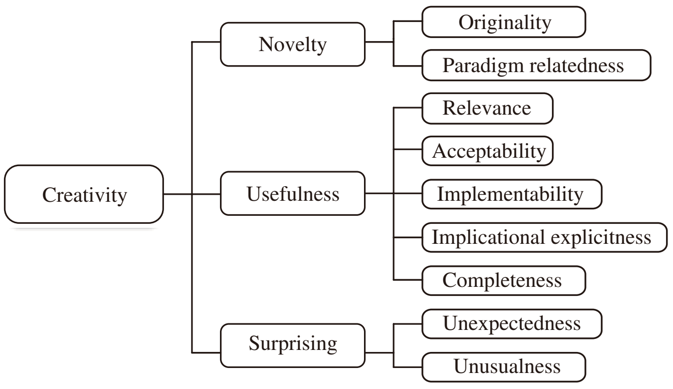


Figure 7 Relationships of creativity metrics and sub-metrics [12].

by using results of one activity as input of another activity. However, it is not necessary to implement all three kinds of activities in every development. That is to say, the practical realities must be worked out in different applications and circumstances.

(c) Idea Evolution. This step is for realising conversion thinking. Among massive amount of initial ideas generated in last step, meaningless ideas are abandoned through verifying language aspects including syntax, semantics and pragmatics for each of them. The remaining ideas are ranked in orders to be represented to the users.

Specifically, as Figure 6 shows, the proposed idea evolution process contains four parts. Firstly, it analyses the generated new ideas via Natural Language Processing (NLP) techniques. That is, an idea can be reviewed from syntactic, semantic and pragmatic views. Then, the ideas are evaluated via certain rules to calculate degrees of creativity. Finally, some of the ideas are provided to users as recommendations assisting on creativity in Internetware.

Dean et al. analysed previous research work and concluded that there are four dimensions for creativity [10, 11], which includes (1) novelty, the degree to which an idea is original, and modifying an existing paradigm; (2) workability, or feasibility, meaning whether the idea can be easily implemented without violating known physical laws; (3) relevance, the degree to which the proposed idea can apply to the stated problem, and solve it; (4) specificity, the degree of clarity, completeness and detail for the idea.

Although the dimensions and sub-dimensions cannot be directly employed in this research, the method used in these studies is worth learning. In particular, some elements are proposed for the creativity of ideas creation [12]. In this paper, the proposed creativity elements are adopted for its own purpose.

Boden [13] suggests an idea can be considered as new and creative only if it contains three kernel features, which are novelty, applicable, and surprising. Specially, it points out that a new idea should be imaginative [12]. Therefore, by adopting some of the method from Dean's studies, Novelty, Usefulness and Surprising forms the creativity metrics for this research. Figure 7 shows the structure of the creativity metrics and sub-metrics.

(d) The Platform. The platform to be built needs to support both individual creators and mass creators to clarify and firm up their ideas, as part of the divergent thinking process. Many of past great

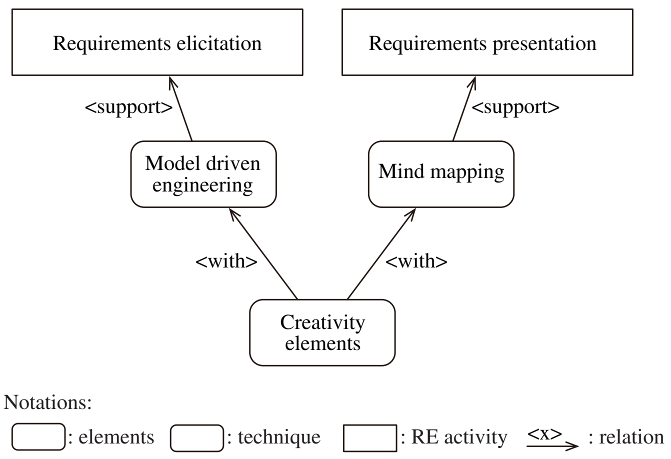


Figure 8 Creativity elements for requirements engineering [12].

inventions in engineering ascribe to individuals, e.g., Alexander Graham Bell, as the inventor of the first practical telephone as well as patent holders of 500 inventions in different areas; Finnish American software engineer, Linus Benedict Torvalds, who was the principal force behind the development of the Linux kernel, and also creator of the revision control system Git as well as the diving log software Subsurface. In the Internetware setting, the prevailing of software crowdsourcing has brought new challenges and opportunities. Stakeholders invite collaborators to further their goals, and extend their space of choices in an Internetware environment. On the basis of such collaboration environments, they fulfill goals by collective contributions that were unachievable otherwise:

- Novelty—possible novelties summarised from past experience/knowledge,
- Usefulness—possible usages summarised from past experience/knowledge,
- Surprises—possible effects summarised from past experience/knowledge,
- User Inputs—at this stage, what a user expects to have and give suggestions to the system, and
- Ideating—a number of ideas generated for the new system to be built.

5 Design evaluation: ranking and recommendation by multi-criteria

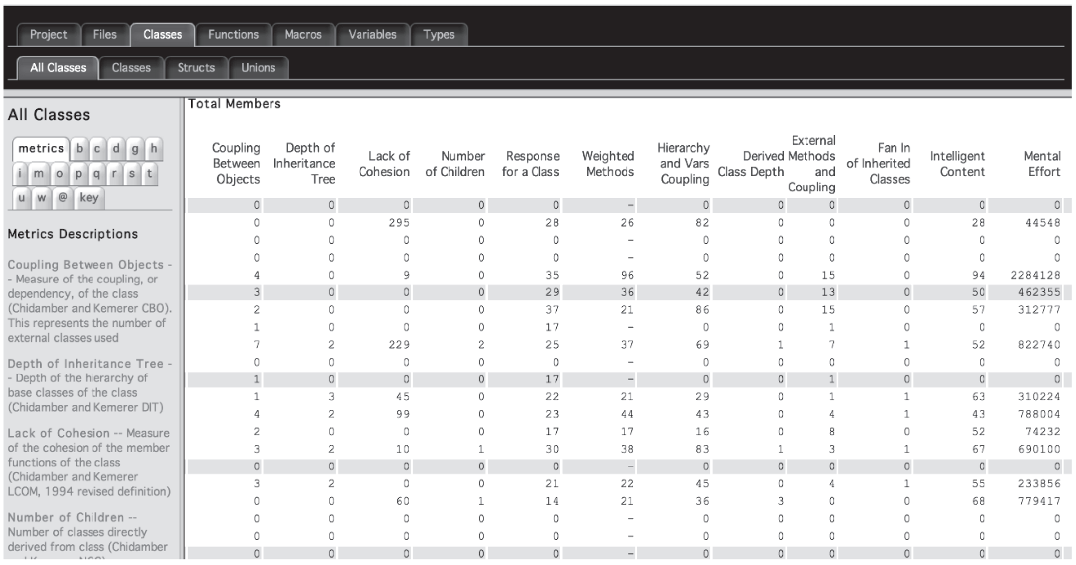
Based on the above criteria, a metrics system is developed in the environment to be built, which will give initial ranking for all the ideas generated and recorded from two previous stages. Here, ideas are evaluated using metrics similar to Halstead metrics in evaluating object-oriented programs, such as Intelligent Content, Mental Effort, Program Difficulty, Program Volume, and so on.

The creativity metrics depicted in Section 4 are adopted to support requirements engineering. Particularly, they are adopted for elicitation and presentation [12]. Figure 8 shows how the creativity metrics work for requirements engineering as creativity elements. In this approach, the creativity elements fit in the ideas evolution step to support the ideas evaluation.

The second part of this stage is to consolidate the user requirements:

Step 1. Model the requirements. In the Internetware environment, requirements determine the functions an Internetware entity provides, e.g., "provide location-based services", and possible expected qualities of services are "with an accuracy of 1 meter". There are many criteria by which alternative candidates are evaluated. System designers and users have their own preferences. Among them, novelty is an emerging attribute to consider. If there are plenty of factors to be handled and these factors are related to each other, correlation analysis is conducted to identify key factors.

Step 2. Evaluate candidates based on knowledge, which contributes to the factor (Figure 9). It is very hard to quantify satisfaction degree for these factors, which are "soft" in nature. Given the weights of factors and requirements satisfaction for each factor, a straightforward and simple way to evaluate the



Metrics Descriptions

Coupling Between Objects -- Measure of the coupling, or dependency, of the class (Chidamber and Kemerer CBO). This represents the number of external classes used

Depth of Inheritance Tree -- Depth of the hierarchy of base classes of the class (Chidamber and Kemerer DIT)

Lack of Cohesion -- Measure of the cohesion of the member functions of the class (Chidamber and Kemerer LCOM, 1994 revised definition)

Number of Children -- Number of classes directly derived from class (Chidamber

Total Members

Coupling Between Objects	Depth of Inheritance Tree	Lack of Cohesion	Number of Children	Response for a Class	Weighted Methods	Hierarchy and Vars Coupling	Derived Class Depth	External Methods and Coupling	Fan In of Inherited Classes	Intelligent Content	Mental Effort
0	0	0	0	0	–	0	0	0	0	0	0
0	0	295	0	28	26	82	0	0	0	28	44548
0	0	0	0	0	–	0	0	0	0	0	0
0	0	0	0	0	–	0	0	0	0	0	0
4	0	9	0	35	96	52	0	15	0	94	2284128
3	0	0	0	29	36	42	0	13	0	50	462355
2	0	0	0	37	21	86	0	15	0	57	312777
1	0	0	0	17	–	0	0	1	0	0	0
7	2	229	2	25	37	69	1	7	1	52	822740
0	0	0	0	0	–	0	0	0	0	0	0
1	0	0	0	17	–	0	0	1	0	0	0
1	3	45	0	22	21	29	0	1	1	63	310224
4	2	99	0	23	44	43	0	4	1	43	788004
2	0	0	0	17	17	16	0	8	0	52	74232
3	2	10	1	30	38	83	1	3	1	67	690100
0	0	0	0	0	–	0	0	0	0	0	0
3	2	0	0	21	22	45	0	4	1	55	233856
0	0	60	1	14	21	36	3	0	0	68	779417
0	0	0	0	0	–	0	0	0	0	0	0
0	0	0	0	0	–	0	0	0	0	0	0
0	0	0	0	0	–	0	0	0	0	0	0

Figure 9 Design evaluation using design metrics (using static analysis tools, e.g., ImageX4D).

overall score is by their weighted sum. Following these steps, we evaluate the satisfaction degree of each candidate Internetware entity.

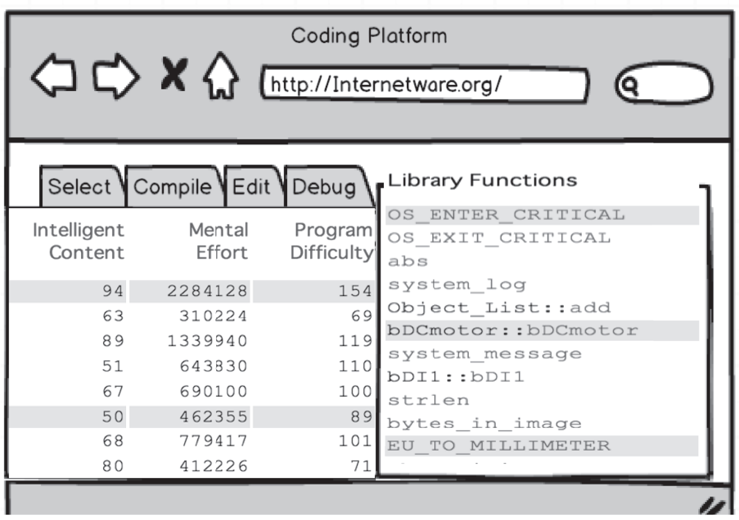
- Evaluating—to evaluate the selected list of ideas
- Ranking—to rank the list of ideas
- User Inputs—user to create one or more ideas based on seeing the evaluation and ranking
- Refining—system to help to refine users' ideas

Then a user evaluates and ranks all ideas again, including newly generated ideas, and generate more ideas until the user is happy with one idea, which will become the specification of the system to be built. Evaluation is crucial at the Internetware paradigm. How to estimate whether a creative design is realizable, and how to judge whether a creative design is a good one. To be creative in developing software in Internetware paradigm, how to effectively involve knowledge from relevant disciplines is very important. Supplying knowledge from these disciplines is part of the tasks for the proposed environment. In a previous study [5], it was identified that Art, Design, Education, Psychology, History and Mathematics are among the first groups of disciplines that are most influential to developing software creatively.

6 Coding and implementation: assets from open-source repositories or crowdsourcing

Coding has been heavily discussed over the years. In the Internetware paradigm, it is strongly suggested that, in addition to writing code, referring to open source codes and coding by crowdsourcing should be used as much as possible (Figure 10). Select—select open-source code such as in GitHub, SourceForge, Programmable Web, etc. Imported code can be Edited, Compiled and Debugged. The following kinds of creativity have been discussed here: exploratory creativity, transformational creativity, and combinational creativity. Exploratory creativity means that the research is conducted in one already existing conceptual space. Transformational creativity means that the research needs to create its own new exclusive conceptual space by transforming an existing conceptual space [14]. Combinational creativity is to combine familiar ideas to shape a new idea, like analogy [15], because, if people want to solve a particular problem, they will think about what already has been held in their hands firstly. It depends on the ability of association.

Users' behavioural data provides important cue for deficiencies in the current software design, and points for improvement in future design. Today's web based software services, after being adjusted by the principles of the Internetware, could collect different kinds of service usage data, which is a good



Intelligent Content	Mental Effort	Program Difficulty
94	2284128	154
63	310224	69
89	1339940	119
51	643830	110
67	690100	100
50	462355	89
68	779417	101
80	412226	71

```
OS_ENTER_CRITICAL
OS_EXIT_CRITICAL
abs
system_log
Object_List::add
bDCmotor::bDCmotor
system_message
bDI1::bDI1
strlen
bytes_in_image
EU_TO_MILLIMETER
```

Figure 10 Coding platform.

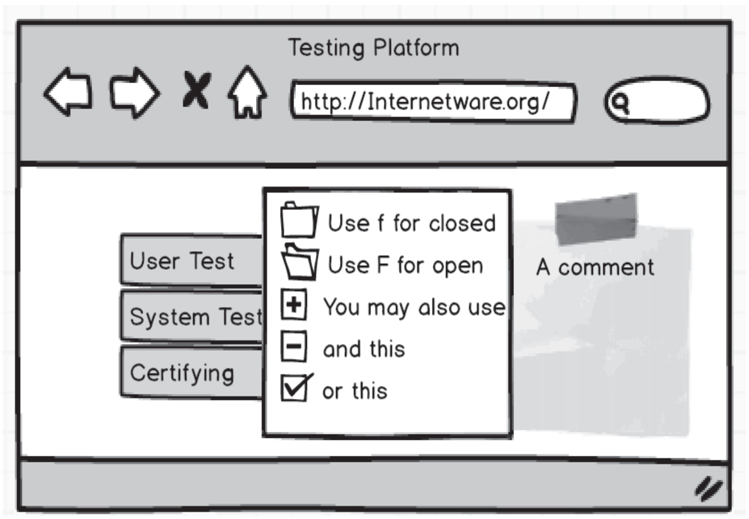


Figure 11 Testing platform.

source of information which helps developers better understand users' behaviour. We explore answers for questions such as: what are the key indicators with regard to the quality of software product design, and how to enhance new product success by understandings obtained from users' behavioural data analysis (Figure 11). In order to fully understand the users' acceptance to the product, the most effective way is to collect as much information as possible about the process of software usage and conduct a deep analysis of them. That is, an end-user testing environment.

7 Case study: software development in E-Health

This section presents a life cycle of software development in E-Health to demonstrate and prove that the proposed method can be applied to support creativity in software development. The project aims to create a single disease website for dentists to manage and research on a large lab test dataset. According to the proposed method, we deployed the method with six steps that were described in previous sections, to achieve the process.

In the Searching step, we found solutions under given context information from reliable sources. We

Table 1 The search result in the Searching step

#	Category	Search result
1	Context	Different types of operation process standard clinical document structures
2	Content	Graphical oral 2D models, 3D navigation models
3	System	E-Health application, HIS software
4	Techniques	.net c# as programming languages, MVC as the software architecture, and OAuth as the security protocol of the platform

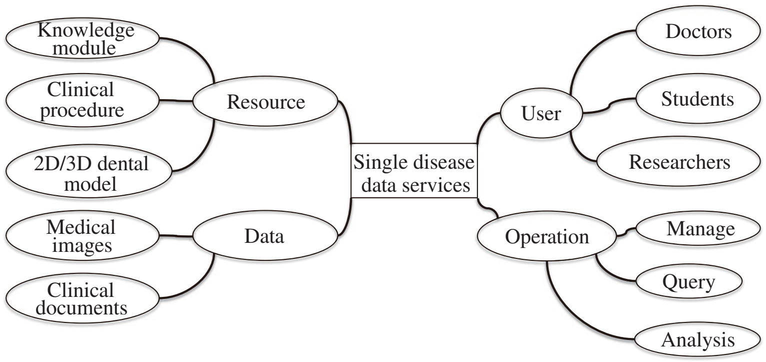


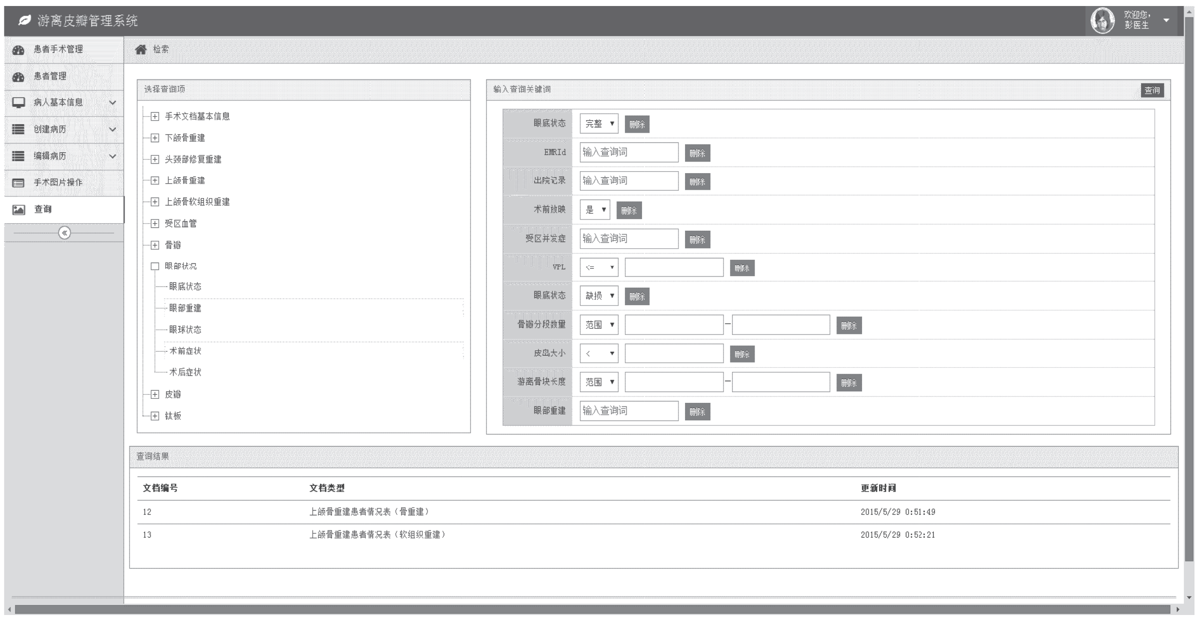
Figure 12 Mind map of single disease service's requirements.

looked for different types of operation process, standard clinical document structures, graphical oral 2D models as the reference resources, and search similar products (such as HIS software) ordered in a list via specific online repositories, and we also looked for some mature techniques, including using .net c# as programming languages, MVC as the software architecture, and OAuth as the security protocol of the platform (Table 1).

In the Ideating step, we make a process of generating, developing and communicating these new ideas. Before the idea generation, we need to build domain knowledge base first, which is the bridge between knowledge collection and creative design. The domain ontology is showed as step 1 in Figure 14, which depends on the resource from Searching step. Base on the knowledge base we start a brainstorm of idea generation from multiple groups of creators, the creators can be a dentist, a stakeholder, a researcher, etc. Using some heuristic rules we can map the ontology to three basis kind of activities which can help to evaluate resource and generate initial ideas. After idea generation, we used algorithms including "Syntax Verification", "Semantic Verification", and "Creativity Ranking", to identify TOP-N ideas. The ranking method is based on some factors: novelty, usefulness, surprises, user inputs and ideation. For example, "data ETL" is strongly recommended because of its usefulness (system needs data) and user inputs (doctors want to migrate data). The main part of generated ideas is shown as step 2 in Figure 14.

In the specifying step, we combine generated ideas and mind mapping techniques for requirements engineering. First we can get the mind map easily by domain knowledge, shown as Figure 12 . Then we need to consolidate the user requirements: (1) we used $i*$ diagram to model the requirements and describe the final goal, functional or non-functional, and find out four factors contributing to customer satisfaction: time, quality, cost, and EMR capability. (2) We then choose AHP to quantify the relative weights of each factors pair. For example, we set $\langle quality, time\rangle = \langle 10, 1\rangle$ because doctors can accept waiting for more accurate answers. (3) We evaluate the satisfaction degree of each candidate entity by these factors and rank the list of them. And we can get some more acceptable ideas as last shown as step 3 in Figure 14.

In the Coding step, we planned to reuse as much open source software assets as possible. We used the keywords, "single disease website", "advanced query", etc., to select on GitHub and SourceForge. From the search results we tested out the three kinds of creativity, and found that a combinational creativity is needed and we also need transformational creativity to create a new exclusive conceptual space because of the lack of advanced query system. Then we use the ImageX4D, which is a static analysis tool mentioned



眼底状态	完整		删除
EMRId	输入查询词		删除
出院记录	输入查询词		删除
术前放映	是		删除
受区并发症	输入查询词		删除
VPL	<=		删除
眼底状态	缺损		删除
骨瓣分段数量	范围	–	删除
皮岛大小	<		删除
游离骨块长度	范围	–	删除
眼部重建	输入查询词		删除

查询结果

文档编号	文档类型	更新时间
12	上颌骨重建患者情况表（骨重建）	2015/5/29 0:51:49
13	上颌骨重建患者情况表（软组织重建）	2015/5/29 0:52:21

Figure 13 System of a free-flap operation data repository.

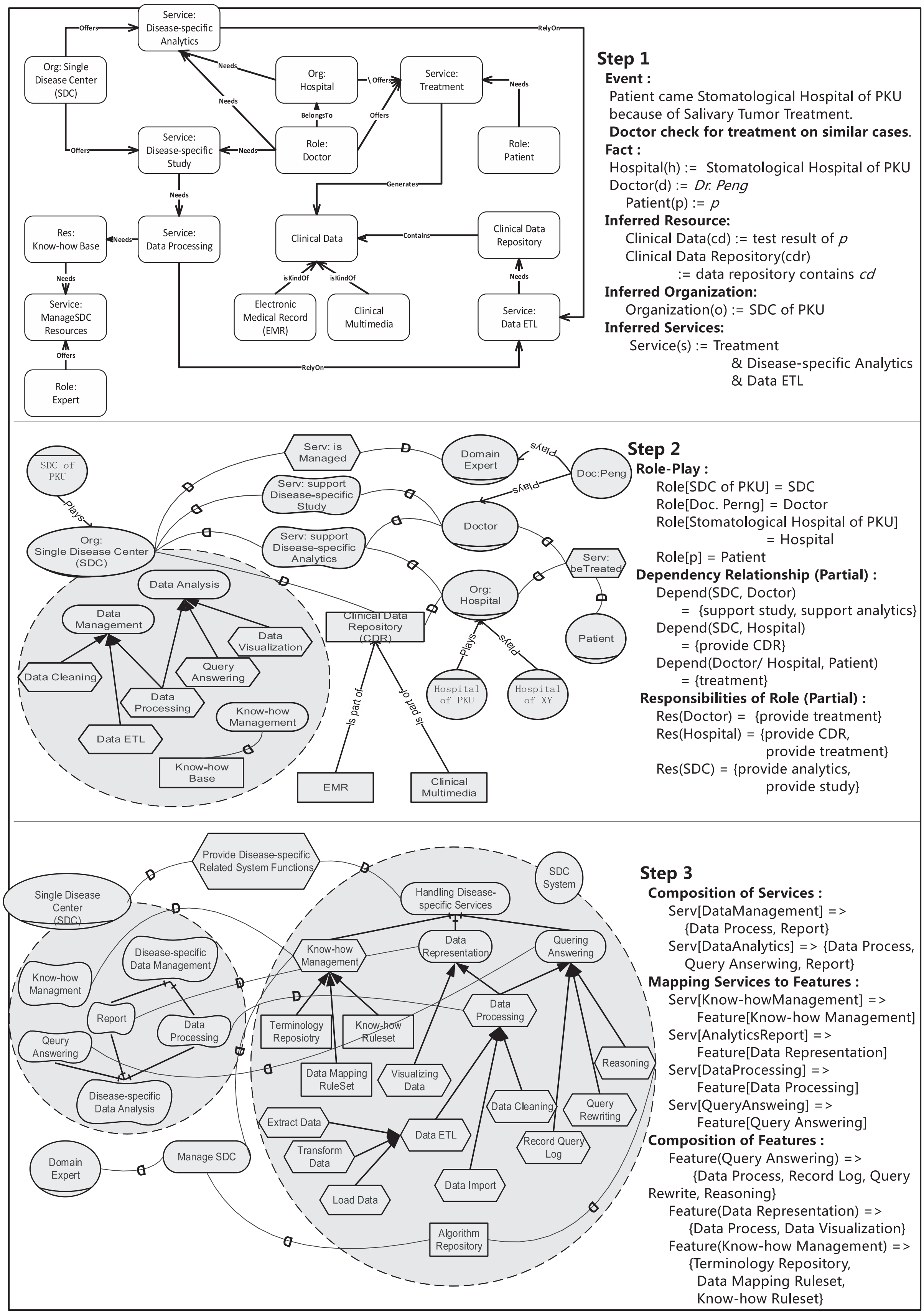
before, to describe the structure of these selected codes and design our system. After the combination of enough mature codes we can start the journey of agile system development.

In the Testing step, we designed an interface for developers to collect service usage data. We used page click logs to understand users' acceptance to the functions; if some pages are visited seldomly, we then go back to Specifying step to revise our functional requirements. We also use users' query log to complete user's profile, which can help designer to change the layout of the system. After a round of testing, we had a good enough software system, which satisfies most users, and then we release it. Through the six phases, the final product is generated semi-automatically as shown in Figure 13.

8 Concluding remarks

In summary, Internetware entities are evaluated from two perspectives: one is fitness for purpose, i.e., users' requirements satisfaction, which involves the satisfaction of all alternative ways of delivering the required functional and non-functional goals. The other is Internetware specific evaluation, which estimates the technical merit of alternatives in achieving a given technical criterion. Among alternative entities, value creation and technical merit are the two aspects to be considered, so satisfaction evaluation and merit evaluation are both necessary. In terms of technical merits of Internetware, there are many preferable characteristics, such as autonomous, collaborative, polymorphic, reactive, evolvable and so on.

The Internetware paradigm is defined by its symbolic features—construction over the Internetwork environment, running on the Internet environment, but the process for software development in it is still shaping in research and practice. This study argues that the Internetware paradigm provides inherent support to software creativity. It is proposed that a non-traditional process is most suitable for this purpose, where up-to-date technologies for searching, end-user programming and end-user testing are combined with modern creativity theories based on existing research experiences, observations and initial experiments. It is also observed that the definition, research scope and challenges of the Internetware and its software process will develop hand in hand. There are a number of open issues yet to be considered in fully implementing the proposed paradigm of Internetware, especially, the integration of mobile computing and Internet; future ways of interactions between computer, software and people; adopting know-how from a variety of interrelated disciplines; evaluation of Internetware products in terms of creativity and usability; dealing with engineering data for Internetware; innovative research methods for the Internetware paradigm, and so on.



Step 1

Event :

Patient came Stomatological Hospital of PKU because of Salivary Tumor Treatment.

Doctor check for treatment on similar cases.

Fact :

Hospital(h) := Stomatological Hospital of PKU

Doctor(d) := *Dr. Peng*

Patient(p) := *p*

Inferred Resource:

Clinical Data(cd) := test result of *p*

Clinical Data Repository(cdr)

:= data repository contains *cd*

Inferred Organization:

Organization(o) := SDC of PKU

Inferred Services:

Service(s) := Treatment

& Disease-specific Analytics

& Data ETL

Step 2

Role-Play :

Role[SDC of PKU] = SDC

Role[Doc. Perng] = Doctor

Role[Stomatological Hospital of PKU]

= Hospital

Role[p] = Patient

Dependency Relationship (Partial) :

Depend(SDC, Doctor)

= {support study, support analytics}

Depend(SDC, Hospital)

= {provide CDR}

Depend(Doctor/ Hospital, Patient)

= {treatment}

Responsibilities of Role (Partial) :

Res(Doctor) = {provide treatment}

Res(Hospital) = {provide CDR,

provide treatment}

Res(SDC) = {provide analytics,

provide study}

Step 3

Composition of Services :

Serv[DataManagement] =>

{Data Process, Report}

Serv[DataAnalytics] => {Data Process,

Query Anserwing, Report}

Mapping Services to Features :

Serv[Know-howManagement] =>

Feature[Know-how Management]

Serv[AnalyticsReport] =>

Feature[Data Representation]

Serv[DataProcessing] =>

Feature[Data Processing]

Serv[QueryAnsweing] =>

Feature[Query Answering]

Composition of Features :

Feature(Query Answering) =>

{Data Process, Record Log, Query Rewrite, Reasoning}

Feature(Data Representation) =>

{Data Process, Data Visualization}

Feature(Know-how Management) =>

{Terminology Repository,

Data Mapping Ruleset,

Know-how Ruleset}

Figure 14 Example case study using the single disease center services domain ontology.

Acknowledgements This paper received partial financial support from Natural Science and Technology Support Program (Grant No. 2015BAH14F02), and National Natural Science Foundation of China (Grant No. 61432020).

Conflict of interest The authors declare that they have no conflict of interest.

References

1 Li Y, Zhou M, You C, et al. Enabling on demand deployment of middleware services in componentized middleware. In: Proceedings of International Symposium on Component-Based Software Engineering. Berlin/Heidelberg: Springer, 2010. 113–129

2 Lv J, Ma X X, Tao X P, et al. On environment-driven software model for Internetware. Sci China Ser-F: Inf Sci, 2008, 51: 683–721

3 Lv J, Ma X X, Tao X P, et al. Research and progress on Internetware. Sci China Ser-E: Tech Sci, 2006, 36: 1037–1080

4 Zheng L W, Tang J, Jin Z. An agent based framework for Internetware computing. Int J Softw Inform, 2010, 4: 401–418

5 Jing D L, Yang H J, Tian Y C. Abstraction based domain ontology extraction for idea creation. In: Proceedings of the 13th International Conference on Quality Software, Najing, 2013. 341–348

6 Jing D L, Yang H J, Shi M Y, et al. Developing a research ideas creation system through reusing knowledge bases for ontology construction. In: Proceedings of the 39th Computer Software and Applications Conference (COMPSAC), Taichung, 2015. 3: 175–180

7 Cook P. The creativity advantage—is your organization the leader of the pack? Ind Commer Train, 1998, 30: 179–184

8 Carroll E A, Latulipe C, Fung R, et al. Creativity factor evaluation: towards a standardized survey metric for creativity support. In: Proceedings of the 7th ACM Conference on Creativity and Cognition. New York: ACM, 2009. 127–136

9 Jellouli I, El Mohajir M. Towards automatic semantic annotation of data rich Web pages. In: Proceedings of the 3rd International Conference on Research Challenges in Information Science, Fez, 2009. 139–142

10 Douglas L D, Jillian M H, Thomas L R, et al. Identifying quality, novel, and creative ideas: constructs and scales for idea evaluation1. J Assoc Inf Syst, 2006, 7: 646–699

11 Puccio G J, Cabra J F. Idea generation and idea evaluation: cognitive skills and deliberate practices. In: Mumford M, ed. Handbook of Organizational Creativity. Amsterdam: Elsevier, 2012. 189–215

12 Jing D L, Zhang C, Yang H J. Using an ideas creation system to assist and inspire creativity in requirements engineering. In: Liu L, Aoyama M, eds. Requirements Engineering in the Big Data Era. Berlin/Heidelberg: Springer, 2015. 155–169

13 Schneider H J, Ehrig H, Pfender M. Graph-grammars: an algebraic approach. In: Proceedings of the 14th Annual Symposium on Switching and Automata Theory. Washington, DC: IEEE, 1973. 167–180

14 Bass L, Clements P, Kazman R. Software Architecture in Practice. Upper Saddle River: Addison-Wesley, 2013

15 Colton S, Pease A, Charnley J. Computational creativity theory: the FACE and IDEA descriptive models. In: Proceedings of the 2nd International Conference on Computational Creativity, Mexico, 2011. 90–95