基于场景自动机的网构软件演化

王茂光 曹怀虎
（中央财经大学信息学院 北京 100081）

摘 要 网构软件是网络开放、动态和多变环境下软件系统基本形态的一种抽象，其演化性要求软件能根据应用需求和运行环境变化而动态演化。引入了软件需求工程中场景的概念，但不同的是不把场景作为一种静态的记录来使用，而是把场景作为一种动态记录来描述软件的动态演化特征。给出了场景的形式化定义及其互补、等价、子集等关系描述，提出了基于场景自动机的网构软件演化方法。软件的性质和行为可以由一系列应用场景来展现，系统的演化通过场景自动机来体现，这为系统自适应演化提供了一种新的解决方法，并支持系统更大粒度的复用。

关键词 网构软件，场景自动机，演化
中图法分类号 TP181 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2014.11.018

Scenario Automation Machine Based Internetware Evolution

WANG Mao-guang CAO Huai-hu
(School of Information, Central University of Finance and Economics, Beijing 100081, China)

Abstract Internetware is an abstraction of software system in the open, dynamic and varying network environment. The internetware evolution characteristic requires the software is capable of evolving dynamically according to the application requirements and running environment changes. The scenario concept in the requirements phase of software engineering domain was introduced to the dynamic evolution. Different from the static requirements description, the scenario is used to illustrate the system dynamic evolution. The scenario formal representation and its complementary, equivalence and subset relations were defined. The internetware evolution method based on the scenario automation machine was illustrated. Thus, the internetware evolution is represented as a series of application scenarios. The system evolution is represented as the scenario transformation. It provides a new method for complex software self-evolution and supports the system large-granularity reuse.

Keywords Internetware, Scenario automation machine, Evolution

1 引言

场景一直有着多角度的解释和使用，其研究最初是从军事领域开始的。近年来，经济学家把场景应用在长期规划上，管理专家把场景应用在战略决策上，政策制定者把场景用在权衡他们行为的结果分析上。目前在IT领域，研究者一般把场景看作是一种能使人机交互和信息系统通信改变的工具。在软件工程领域，场景常用于描述接口规范的人机交互工程，或用来阐述软件系统的需求。软件工程师通常把场景研究作为一种有效的方法来发掘用户的需求，并把需求更好地植入到系统开发工作流程中。

网构软件强调的是以软件构件等技术支持的软件实体以开放、自主的方式存在于Internet的各个节点之上，任何一个软件实体可在开放的环境下通过某种方式加以发布，并以各种协同方式与其它软件实体进行互连互通[1]。网构软件具有自适应、动态协同、开放、演化和环境感知的特性。演化性指的是网构软件能根据应用需求和运行环境变化而动态演化，主要表现在其实体元素数目、行为和系统功能的可变性等。

本文引入场景的出发点主要是支持网构软件自适应环境的变化，具有动态演化的特性。目前研究软件演化的方法主要利用Petri网等形式化方法和体系结构等质量属性描述方法[2,3]。但其存在着两大挑战：

1）软件局部自主性和全局协调性的矛盾。即往往不存在集中式的控制中心来协调构件的适应性行为。

2）设计的预先性和运用的不可预知性的矛盾。即可复用的构件的实现应该独立于特定的应用环境，不可能在自主构件中预先定制对各种应用环境的自适应能力。

为了应对上述挑战，在分析各领域场景研究的基础上，我们提出了一种基于场景的自主构件演化解决方案，其设计思路如下：

首先，软件系统的性质和行为由一系列应用场景来展现，场景规定了构件的任务、行为和适用策略等，随着场景的变

到稿日期：2013-12-09 返修日期：2014-02-09 本文受国家自然科学基金(61073020)，国家863高技术研究发展计划项目(2009AA01Z139)，国家留学基金委资助。

王茂光(1974—)，男，博士，副教授，硕士生导师，主要研究领域为多Agent系统、智能软件、软件工程等，E-mail：wangmg@cufe.edu.cn；曹怀虎(1977—)，男，副教授，硕士生导师，主要研究领域为计算机网络、社会网络。

迁，构件的行为及其之间的交互也随之自动发生变化，从而体现网构软件对不同应用场景的适应性。

其次，在系统实现的过程中，将场景实体化为特殊的构件，场景构件把其它构件作为依赖服务或交互对象，避免了集中式控制。通过动态部署场景，可以实现网构软件对未预定义的应用场景的动态适应。

本文第 2 节介绍了场景相关工作；第 3 节描述了场景的定义、关系及场景自动机；第 4 节描述了基于场景的演化方法和应用实例；最后总结全文。

2 相关工作

在软件和系统工程领域，特别强调需求工程中场景的研究和实践。很多学者都给出了场景的定义，包括从对现实世界的描述到模型和需求规约，但到目前为止场景并无一个统一的定义。代表性的工作如 P. A. Gough[4] 和 J. M. Carroll[5] 将场景看作是用自然语言、图片或其他媒介表述的现实世界，而 A. Cockburn[6] 和 A. G. Sutcliffe[7] 将场景看作是模型，例如用例、用例中的路径以及其它事件序列的描述。

由德国、英国、法国、比利时学者共同参与的 CREWS (Cooperative Requirements Engineering With Scenarios)项目对场景做了深入的研究，建立了丰富的理论基础。其中，M. Jarke 等人[8]从人机交互、软件工程与系统工程以及战略管理等 3 个研究领域对场景的定义、建模、使用和评估等问题进行了系统的阐述。C. Rolland 等人[9]从内容、形式、目的以及生命周期 4 个维度出发，建立了场景的分类框架。CREWS 项目的一个重要贡献是其对现状和期望做了显式的区分，并认为基于场景的现状和期望建模能够更有效地促进利益相关者形成对现状和期望的共识，从而实现从现状到期望的顺利过渡。

在传统的方法中场景是作为一种静态的记录来使用的。上述工作给场景的定义各不相同，但都将场景作为一种静态记录来描述现实世界的业务、用户和系统的交互、系统的行为等。目前场景研究中存在的问题有：1)缺乏明确的界定目标和清晰的结构；2)缺乏合适的划分场景的标准。较低粒度的场景具有较高的抽象概括水平，易建立但在实际应用中复杂度会较高。相反，高粒度的场景可能包含不太准确的假设信息，因此，常常需要检验和修正场景。

3 场景的定义及场景自动机

定义场景是系统在一定的背景下，对相关任务、参与构件和适用策略的一种有目的的描述。定义场景 $Scenario=(ID,R,C,A,B,P)$，其中：

ID={name,triggers}，表示场景的主键集，用来唯一标识场景，也可根据需要灵活扩充。其中元素分别表示场景的名字、触发条件；

$R=\{r_1,r_2,\cdots,r_s\}$，表示场景中的角色集，角色是具有一定职责和能力的抽象实体，承担某个角色的构件则成为该角色的一个实例，即角色可以看作是一组具有相同能力构件的集合；

$C=\{s_1,s_2,\cdots,s_M\}$，表示场景的上下文，s_i 表示环境的状态；

$A=\{a_1,a_2,\cdots,a_K\}$，表示场景的参与者，a_i 表示参与的构件，常与具体的角色相绑定。具体角色的绑定、继承和演化可参考文献[10]；

$B=\{b_1,b_2,\cdots,b_L\}$，表示构件的行为集，可以描述场景的具体情节；

$P=P_g\cup P_a\cup P_u$，表示适用于场景的策略集，包括目标策略 P_g、动作策略 P_a 和效用策略 P_u 等，分别表示要实现的目标、执行的动作规则和执行后的效果。

进而，考虑到场景粒度的划分及设计过程中场景的分割、合并等操作，定义如下场景的关系：

1. 互补关系：$Com(S_i,S_j)$，

1) $\forall R\in S_i,R\in S_j$

2) $\exists C\in S_i,C\in S_j$

3) $\exists A_i\in S_i,A_i\in S_j$

4) $\exists P_g\in S_i,P_g\in S_j$

反之亦然，满足对称性。互补关系描述的场景遵循一个共同目标策略，并在角色上保持一致性。例如两个场景实现一个共同目标“满足获得博士学位的要求”，两个场景分别是学习课程和论文答辩。基于该关系，这两个场景可以合并为一个大的场景“获得博士学位”。

2. 场景的等价关系：$Eqv(S_i,S_j)$

1) $R_i\Leftrightarrow R_j$

2) $C_i\Leftrightarrow C_j$

3) $\exists A_i\in S_i,A_i\in S_j$

4) $B_i\Leftrightarrow B_j$

5) $P_i\Leftrightarrow P_j$

反之亦然，满足对称性。等价关系的场景遵循共同策略，并在角色、上下文、行为和策略上保持一致性，等价场景可以直接复用。

3. 场景的包含关系：$Inc(S_i,S_j)$

1) $C_i\subseteq C_j$

2) $R_i\subseteq R_j$

3) $\exists A_i\in S_i,A_i\in S_j$

4) $B_i\subseteq B_j$

不满足对称性，满足传递性。场景的包含关系表示一个场景是另一场景的子集，该关系可以用于场景的粒度划分。

4. 场景的前件关系 $Pre(S_i,S_j)$

1) $\exists C\in S_i,C\in S_j$

2) $\exists A_i\in S_i,A_i\in S_j$

3) $\exists B_i\in S_i,B_i\in S_j$

不满足对称性，满足传递性。前件关系在场景的上下文、参与构件和行为中得到体现。前件关系允许考虑场景的时序关系，能够定义场景的时序序列。

基于场景的定义，定义场景自动机 SA 为一个五元组，$SA=(S,E,\delta,S_0,S_f)$，其中：

S：场景的非空集；

E：触发场景转换的事件集；

δ：场景转换策略，是 $S\times E\rightarrow 2^S$ 的映射，$\delta(S_i,e)=\{S_j,S_k,\cdots,S_n\},S_j,S_k,S_m\in S$；

S_0：初始场景集，$S_0\subset S$；

S_f：终止场景集，$S_f\subset S$。

其中，简单的线性场景转换策略如 $\delta=S_0 \xrightarrow{E_{0\to i}} S_i \xrightarrow{E_{i\to j}} S_j \cdots \xrightarrow{E_{\cdots m}} S_m$，通过一个事件序列 E 构成了场景的转换序列。以如下 3 个场景为例，应用场景转换如图 1 所示。

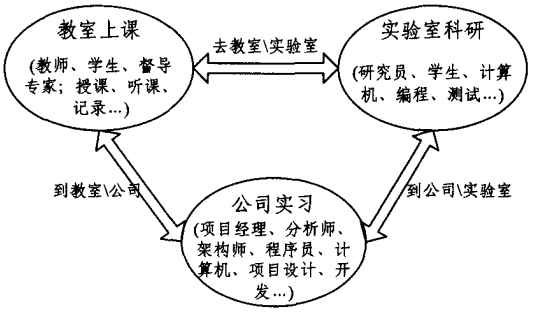

图 1　场景的转换

场景集合 Scenario＝{教室上课，实验室科研，公司实习}，具体如表 1 所列，该场景自动机的五元组也一目了然。

表 1　场景描述

	场景 1	场景 2	场景 3
ID	Sce_Class_Study	Sce_Lab_Research	Sce_Company_intern
R	{教师，学生，督导专家}	{课题负责人，学生}	{项目经理，分析师，架构师，程序员}
C	{教室，黑板，桌椅，多媒体设备}	{实验室，5 台计算机}	{公司开发部，15 台联网计算机}
A	{1 名教授，30 名学生，1 名督导专家}	{1 名研究员，1 名博士生，3 名硕士生}	{1 名项目经理，1 名分析师，5 名程序员}
B	{授课，听课，提问，讨论}	{系统分析，架构设计，编程，测试}	{讨论，编码，测试}
P	{开始时间 8:00 and 结束时间 8:50}∪{授课学时内容讲完}∪{教学效果良好}	{设计方案经过审核 and 子系统通过测试}∪{完成指定科研任务}	{完成预计开发工作量}∪{完成项目实习报告}
E	到教室上课↔到实验室做科研↔到公司实习		

4　基于场景自动机的系统演化

构件在不同的场景中的行为方式将发生变化，场景自动机实际上定义了构件如何适应不同的应用场景，随着场景的变迁，整个系统的功能也随之演化[11]。

4.1　演化设计方法

基于场景自动机的系统演化设计如下：

1)系统的性质和行为由一系列应用场景来展现，系统的演化通过场景自动机来体现，当发生特定的事件时，应用场景会发生变化，即变迁到新的应用场景，场景的变迁规定了构件如何适应不同的应用场景。

2)应用场景的变迁由特定的事件触发，当变迁发生时，向后继应用场景发送消息，通知后继场景进入运行状态，并进而对构件的行为进行新的约束。

自适应软件可以由一系列应用场景来展现，设计时如图 2 所示。

在软件动态演化的过程中，场景关注构件的行为、交互和策略。场景从更大粒度上描述了系统构件之间的相互作用、角色演化和行为变迁，并提供了复用性。设由场景自动机映射到具体的软件系统为 (S,E,δ,S_0,S_f)。其中，S 表示系统中的场景集合，E 为相关应用在场景间控制转换的事件集，设 $E_{i\to j}$ 反映了应用执行场景 S_i 转至 S_j 执行的事件(集)，$(S_i, E_{i\to j}, S_j)\in\delta$ 构成了一个具体的场景转换，S_0 为应用执行的起始场景，S_f 为应用执行的终止场景。

在系统实现过程中，构件进入不同场景时，扮演不同的角色，并具有不同的行为和策略，策略是对其行为的具体约束，则随着场景的变迁(如构件发送消息给场景构件)，进入场景的构件也随之发生变化，并且通过动态部署场景构件，可以使自主构件适应各种场景。

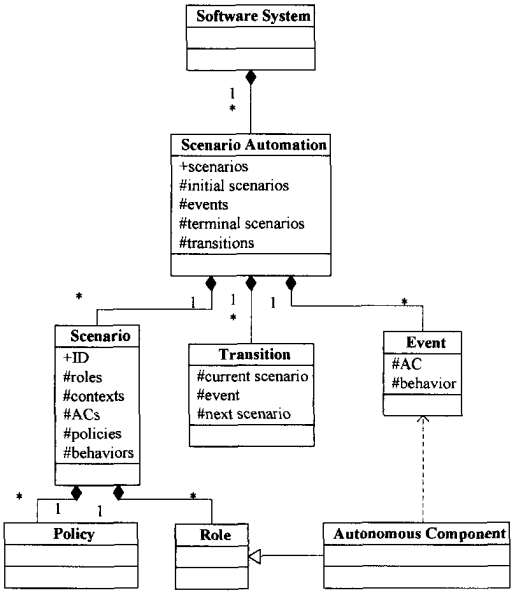

图 2　基于场景的软件系统

4.2　智能交通实例设计

以智能交通系统为例，一个路面复杂、车辆众多的交通系统，若从单个车辆和路来考虑，有很多种状态出现。若从场景来考虑，基本可以分为交通正常、行驶缓慢、交通拥堵、交通事故、设施故障、交通管制等几大场景，通过预定义这些场景，可以指导该场景下的车辆和信号灯、交警的行为，提供场景粒度级的系统优化。整个系统自适应环境的演化具体体现在车辆和信号根据所在场景和自身策略，执行自身行为来适应环境的变化。基于整个系统的场景自动机，整个复杂系统体现出动态演化性，如图 3 所示。

场景中主要构件如下：

1)车辆

接口：进入场景、离开场景。

数据项(环境状态)：道路状况(拥塞、路滑等)，所在位置。

交互构件：道路、交通台、导航仪等。

决策：当收到车辆询问是否同路的信息时，调用应答车辆是否同路；当收到请求 coupling 的信息时，把该车辆加入 coupling 列表；当收到交通台关于道路拥塞等的信息，调用路由操作；当数据项道路状况变为拥塞、路滑等，调用相应的行驶模式操作；每当做出某个具体驾驶操作的时候，向 coupling 列表中的车辆发信号；当检测到有路标，向道路发送询问路标信息等。

行为包括：

切换：将当前道路从交互列表/依赖服务中删除，把下一个要进入的道路加入交互列表/依赖服务，并发送进入环境消息；

应答车辆是否同路：调用导航仪提供的服务，决定是否同路；

路由：调用导航仪，返回下一个路口或道路。

2)导航仪：与车辆1对1绑定

接口：返回当前位置到目的地的距离；返回下一个路口或下一条道路；返回是否途径特定的路口或道路。

3)交通台

接口：返回交通状况信息(拥堵、事故、管制等)。

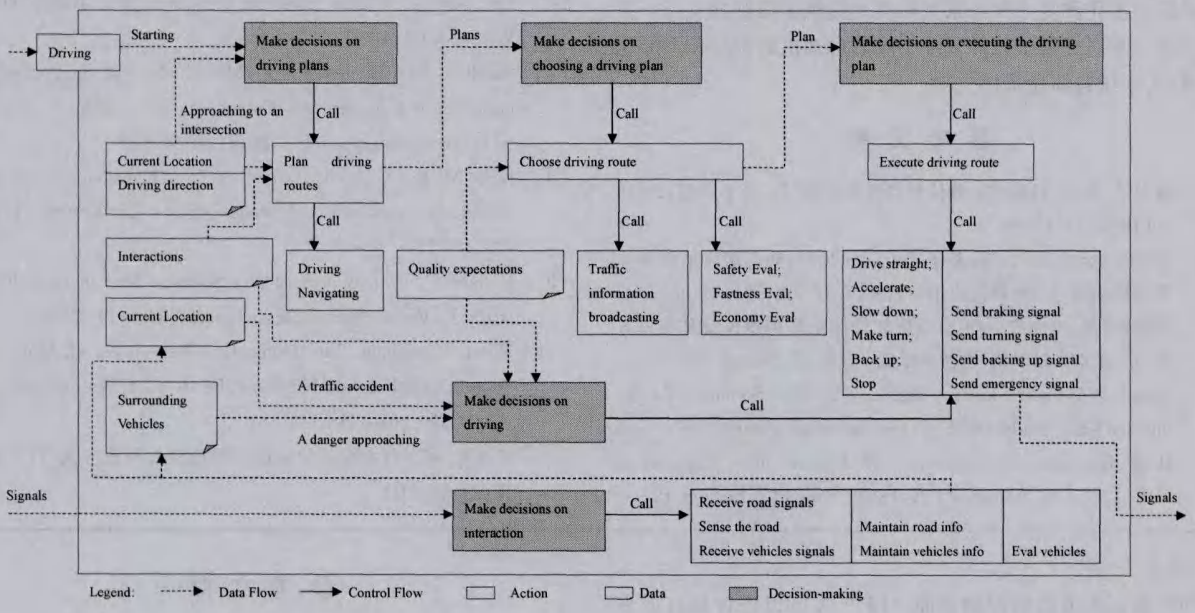

图3 智能交通示意图

4)道路

数据项：入口处道路名、出口处道路名、入口处交通灯、出口处交通灯、路标信息、位置、长度、车道数(上行、下行)、限速、路上车辆数。

决策策略：当收到车辆进入消息，执行车辆进入行为；当收到车辆离开消息，执行车辆离开行为；当车辆违规，做记录；当救护车进入，道路异常(拥堵、有冰、施工)等，通知其中所有车辆。

行为：关闭车道、打开车道、车辆进入、车辆离开、广播信息操作等。

构建系统的场景自动机如图4所示。

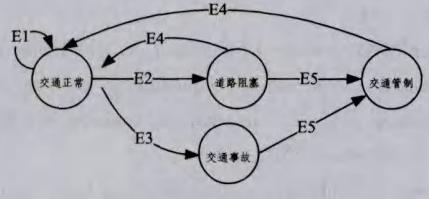

图4 智能交通场景

其中事件集 $E=\{E_1,E_2,E_3,E_4,E_5\}$，表示触发场景变迁的事件，如交通广播、行驶速度、警示牌等，具体设计时要分别定义实现上述构件、场景和适用策略，在此不展开细述。

实验中，我们模拟了正常行驶、交通事故、交通管制、道路阻塞4个场景，而且系统支持动态部署新的场景。在系统中，首先车辆处于正常行驶状态，一旦道路阻塞或交通管制等新的场景出现，车辆会依据场景自动机做出决策调整自身的行为，如调整车速、改变行驶路线等。

当交通阻塞时，场景自动由正常行驶转换为交通阻塞场景，则上述构件的相关属性及其行为和策略集随之自动变化，如图5所示。

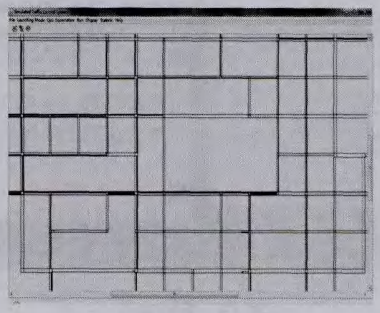

图5 交通堵塞场景变迁

此时评估不同场景下构件的质量属性(到达时间)的变化，随着路上车辆的增多，车辆在动态执行行为和策略的过程中，平均速度也动态下降或上升。汽车平均速度和拥堵率如图6所示。

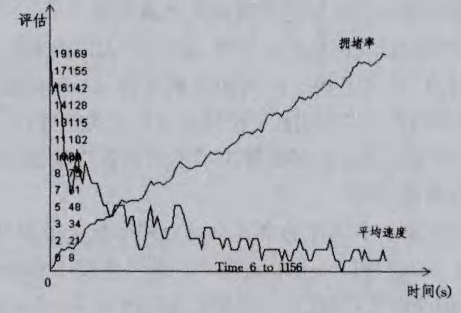

图6 系统的质量属性评估

实验的设计主要验证基于场景自动机的构件和整个系统的演化性，表明构件能根据场景变迁自动改变行为和调整行车路线。设计中合理地划分和定义场景，通过设计场景自动机来支持整个系统的演化过程。

结束语 本文给出了场景及其关系定义，研究了场景在软件演化中的作用。本文重点研究基于场景自动机的软件系

统动态演化过程。系统的性质和行为的动态演化通过应用场景的变迁来体现，当发生特定的事件时，应用场景会变迁到新的场景，构件在不同的场景下的行为方式也随之发生变化，这为系统的演化和复用在场景大粒度上提供了很好的支持。我们将进一步研究复杂软件系统中场景自动机的设计和图形化建模，为系统自演化提供支持。

参 考 文 献

[1] 杨芙清，梅宏，吕建，等. 浅论软件技术发展[J]. 电子学报，2002，30(12A)：1901-1906

[2] 谢仲文，李彤，代飞，等. 基于 Petri 网的面向动态演化的软件体系结构建模[J]. 计算机应用与软件，2012，29 (10)：36-39

[3] 冯耀东，黄罡，梅宏. 一种自适应软件体系结构建模及其实施方法[J]. 北京大学学报：自然科学版，2008，44(1)：：67-76

[4] Gough P A，Fodemski F T，Higgins S A，et al. Scenarios：An Industrial Case Study and Hypermedia Enhancements[C]// 1995 IEEE International Symposium on Requirements Engineering (RE'95). Los Alamitos CA：IEEE Computer Society Press. 1995：10-17

[5] Carroll J M. Making Use：Scenario-Based Design of Human-Computer Interactions[M]. Cambridge MA：MIT Press，2000

[6] Cockburn A. Writing Effective Use Cases [M]. MA：Addison-Wesley，2001：159-176

[7] Sutcliffe A G，Maiden N A M，Minocha S，et al. Supporting Scenario-Based Requirements Engineering[J]. IEEE Transactions on Software Engineering，1998，24：1072-1088

[8] Jarke M，Bui X T，J M. Carroll Scenario management：An interdisciplinary approach[J]. Requirements Engineering，1998，3 (1)：155-173

[9] Rolland C，et al. A proposal for Scenario Classification Framework[J]. Requirements Engineering，1998，3(1)：23-47

[10] Wang Mao-guang，Mei Hong，Jiao Wen-pin，et al. Multi-agent System Collaboration Based on the Relation-Web Model[C]// AICI 2010. 2010：132-144

[11] 王茂光. 基于自主构件的自适应网构软件开发方法[D]. 北京：北京大学，2011

(上接第 73 页)

人网中建议的项目所在的页面。用户 A 可以进行预订或者放弃。该过程如图 13 所示。

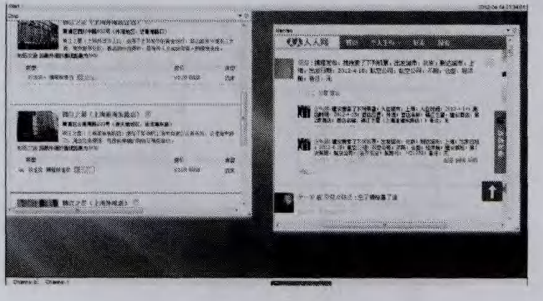

图 13 采纳建议

结束语 在 Web 2.0 时代，富客户端应用兴起。混搭可以解决同一个富客户端的富客户端应用之间的信息流通，但是，除非应用本身具有服务器账户间的直接信息传递机制，否则 mashup 对于不同富客户端间的富客户端应用的信息传递仍然无能为力。因此，一般情况下，用户无法获得他人经验或集体智慧的帮助而更好地使用富客户端应用。

本文利用社交网络的广泛性、交互性、及时性和汇聚集体智慧的特点，提出了基于社交网络服务的 mashup 连接子的设计，使得用户可以利用社交网络好友的人类智慧，更充分地使用富客户端应用，同时也解决了不同富客户端上的应用之间的信息传递问题。

同时在基于中间件的富客户端应用 iMashup 框架内，实现了基于人人网的 mashup 连接子。通过连接子和携程网的 mashup 构件组装，完成了一套基于人人网的 iMashup 连接子的携程应用场景，验证了连接子设计的正确性。

目前利用人人网的状态发布机制进行信息传递，将来可以利用更加丰富的传递形式，如照片、日志、站内信等形式，更加充分地利用他人经验和集体智慧。

可以开发更多的富客户端构件，与基于社交网络的 mashup 连接子进行组装，实现更为复杂的应用场景。

参 考 文 献

[1] Qi Zhao，Gang Huang，Huang Ji-yu，et al. An on-the-fly approach to web-based service composition[C]// Congress on Services Part II. Beijing，China，2008：208-209

[2] Liu Xuan-zhe. Composing data-driven service mashups with tag-based semantic annotations[C]// Proceeding of the IEEE International Conference on Web Services(ICWS). Washington，DC，USA，2011：243-250

[3] Wijesiriwardana C，Ghezzi G，Gall H. A guided mashup framework for rapid software analysis service composition[C]// 2012 19th Asia-Pacific Software Engineering Conference (APSEC). Hong Kong，China，2012：725-728

[4] Kim J S，Yang M H，Hwang Y J，et al. Customer preference analysis based on sns data[C]// 2012 Second International Conference on Cloud and Green Computing (CGC). Hunan，China，2012：609-613

[5] Sachan M，Contractor D，Faruquie T A，et al. Using content and interactions fordiscovering communities in social networks[C]// Proceedings of the 21st international conference on World Wide Web. Lyon，France，2012：331-340

[6] Kukreja N. Winbook：a social networking based framework for collaborative requirements elicitation and winwin negotiations [C]// 2012 34th International Conference on Software Engineering (ICSE). Zurich，Switzerland，2012：1610-1612

[7] 赵祺. 富客户端运行环境自适应中间件研究[D]. 北京：北京大学信息科学技术学院软件研究所，2012

[8] 杨杰. 基于软件体系结构的网构软件组装技术研究[D]. 北京：北京大学信息科学技术学院软件研究所，2007

[9] A mashup tool for creating personalized，web-based and service-oriented mashup systems in the way of WYSIWYG [OL]. [2013-05-30]. http：//www. github. com/sakinijino/imashup

[10] 陈一舟. 人人网用户量今年将达 2 亿[EB/OL]. [2012-02-14]. http：//www. techweb. com. cn/it/2012-02-14/1151875. shtml