一种基于问题框架的软件密集型系统增量需求问题求解方法

董瑞志[1),2)] 彭 鑫[1)] 赵文耘[1)]

[1)](复旦大学计算机科学技术学院 上海 201203)

[2)](常熟理工学院计算机科学与工程学院 江苏 常熟 215500)

摘 要 软件密集型系统的增量需求是在原有需求基础之上,以增量、迭代的方式提出的新需求.在增量需求分析及求解过程中,如何在复用现有解决方案的基础上诱导出增量需求的解决方案,如何验证增量需求解决方案的正确性,如何评估增量需求解决方案的潜在风险,就成了必须解决的重要问题.文中把增量需求解决方案的求解问题视为一个典型的软件开发问题——增量需求问题,并据此提出一种基于软件开发问题框架的增量需求问题求解方法.该方法采用增量、迭代的方式求解增量需求的解决方案,把解决方案的验证问题转换成可满足问题进行求解,同时复用特定关注点知识评估解决方案的潜在风险以支持后续的设计决策.文中通过一个基于科技助老系统的案例研究,展示了应用该方法求解增量需求解决方案的过程,验证了方法的有效性.

关键词 软件开发问题框架;需求建模;需求分析;软件密集型系统

中图法分类号 TP311 **DOI 号** 10.3724/SP.J.1016.2014.00551

Problem Frames-Based Approach for Finding Solutions of Incremental Requirements within Software-Intensive Systems

DONG Rui-Zhi[1),2)] PENG Xin[1)] ZHAO Wen-Yun[1)]

[1)](*School of Computer Science, Fudan University, Shanghai* 201203)

[2)](*School of Computer Science and Engineering, Changshu Institute of Technology, Changshu, Jiangsu* 215500)

Abstract The requirements of software-intensive systems are increasingly developed in an incremental and iterative manner. During the analysis of incremental requirements, the considerable issues are how to find solutions for incremental requirements by reusing the components within the current solutions, how to argue the correctness of the derived solutions which have no after-effect on the satisfaction of the implemented requirements, and how to evaluate the possible risks of the newly derived solutions so as to support the follow-up design decisions. To address such challenges, we look on the issue of finding solutions for incremental requirements as a typical software development problem (namely, the problem of incremental requirements), and then propose a problem frames-based method. Our approach takes an iterative and incremental process to find solutions for incremental requirements, to validate the correctness of the newly generated solutions by the application of SAT based techniques, and to reuse the knowledge of typical concerns to identify possible risks within the solutions in order to make preparation for the follow-up design decisions. We demonstrate our method on a case study of a smart nursing home system. Experimental results provide guidelines on how to use our method, and show the effectiveness of our approach.

Keywords problem frames; requirement modeling; requirement analysis; software-intensive system

收稿日期:2012-07-06;最终修改稿收到日期:2013-11-13.本课题得到国家"八六三"高技术研究发展计划项目基金(2013AA01A605)、教育部博士点基金(20100071110031)资助.**董瑞志**,男,1980 年生,博士研究生,主要研究方向为软件开发问题框架、软件模型检测.E-mail:nature_dong@126.com.**彭 鑫**(通信作者),男,1979 年生,博士,副教授,主要研究方向为自适应软件系统、软件再工程、软件产品线.E-mail:pengxin@fudan.edu.cn.**赵文耘**,男,1964 年生,教授,研究领域为软件工程、电子商务.

1 引 言

目前,越来越多的软件产品以创新型的软件密集型系统(Software-intensive System)的形式推出并占领市场.其需求工程已成为一个贯穿软件生存周期、跨越项目和产品边界的持续过程[1-2].在软件密集型系统的需求工程过程中,软件需求往往以增量、迭代的方式在现有需求基础上进行诱导和分析.例如,在现有产品基础上根据用户反馈、市场分析增量确定下一版本产品需求,或根据差异化使用环境和用户需求确定变体产品需求.在增量需求诱导及分析过程中,需解决如下问题:(1)分析现有解决方案,识别其中可复用的软件规约,进而诱导出增量需求的解决方案;(2)检验增量需求解决方案的正确性,保证增量需求解决方案能够满足增量需求,同时对已有需求不造成任何负面影响;(3)识别并评估增量需求解决方案的潜在风险,为后续设计决策提供支持.

软件密集型系统中需求的满足是软硬件解决方案及相关上下文要素共同作用的结果.因此,求解软件密集型系统增量需求问题时需要综合考虑上下文环境、解决方案和需求三者之间的关系.软件开发问题框架[3]强调对软件系统将要作用的上下文环境的刻画,指出需求工程的任务是开发出在给定环境中能够满足涉众需求的软件产品,强调通过复用已有软件开发问题的求解经验(即问题框架)诱导出给定的软件开发问题的解决方案.因此,软件开发问题框架可以为增量需求问题的求解提供系统化的指导.

在需求工程领域,软件开发问题框架方法已经被用于分析软件需求的定制、复用和演化[4-6].以软件开发问题框架为基础,Dao 等人[4]使用特征模型描述软件产品族中需求的可变性,利用软件开发问题框架方法指导特征模型的定制及软件资产的复用来满足不同用户的需求,但未涉及特征(需求)演化问题.Ernst 等人[5]关注需求演化问题的求解,从用户需求、软件解决方案和给定领域三者之间的因果关系出发,提出面向目标的增量需求解决方案诱导方法.Ernst 等人强调在已经明确增量需求问题的情况下诱导出解决方案,并未涉及如何分析及明确增量需求问题的问题结构.Tun 等人[6]关注软件密集型系统软件产品线中需求(/特征)演化问题.他们把已实现的特征作为可复用的基本单元,通过引入表示增量需求解决方案的包装机器来满足增量需求,使用事件演算器分析增量需求解决方案的正确性,但并未关注解决方案的风险分析问题.

鉴于此,本文提出一个基于软件开发问题框架的软件密集型系统增量需求解决方案求解方法.该方法利用基于问题框架的需求分析技术,抽取出现有需求及其解决方案作为增量需求分析的基础.针对涉众提出的增量需求,采用增量、迭代的分析过程诱导出增量需求的解决方案.该方法还将增量需求解决方案的验证问题转换成可满足问题(Propositional Satisfiability Problem)[7],使用可满足性(Satisfiability,SAT)求解器检验增量需求解决方案的正确性.一旦发现该解决方案无法满足增量需求,则进入一个新的迭代过程,重新求解增量需求的解决方案.重复上述过程,直至得到正确的增量需求解决方案.此外,该方法还复用特定关注点知识,识别出增量需求问题解决过程的潜在风险,为增量需求解决方案的完善及后续设计决策提供支持.

本文第 2 节介绍背景知识;第 3 节给出增量需求问题的定义;第 4 节描述本文提出的增量需求解决方案的诱导方法;第 5 节介绍应用该方法求解科技助老系统中增量需求解决方案的过程,展示了方法的有效性;第 6 节描述相关工作;最后,总结全文并展望下阶段的研究工作.

2 背景知识

本节首先介绍软件开发问题框架的基本概念和思想,然后介绍 Alloy 语言及其建模分析工具 Alloy Analyzer.

2.1 软件开发问题框架

软件开发问题框架[3]强调对软件将要作用的环境进行刻画,使用需求、软件和上下文环境之间的因果关系来描述软件开发问题.在该方法中,求解一个软件开发问题就是要诱导出一个在给定的环境 W 中,能够满足用户需求 R 的解决方案 S,即一个软件开发问题可以表示成式(1):

$$W,S\vdash R \tag{1}$$

软件开发问题框架方法使用问题图刻画软件开发问题.二元符号"$\vdash$"表示其左件元素的满足将导致右件元素的满足,即如果给定的现实世界 W 中要素满

足其领域假设，机器 M 满足机器规约，将使得需求 R 得到满足.

例如，一个利用信息技术服务于老年人日常起居的科技助老系统现有解决方案仅提供了一键拨号服务. 借助该服务，独居老人可以按下老人手机上的求助按钮与社区呼叫中心连接，社区呼叫中心系统将同步显示呼叫者的姓名、年龄、地址、身体状况、关爱员(如老人的亲属或保姆等)的联系方式等信息. 社区服务中心的值班人员了解到老人的紧急情况后，可以第一时间通知老人关爱员予以处置. 可以用图 1 所示的问题图表示一键拨号服务所关联的软件问题(基于篇幅的限制，省略了领域现象的详细描述). 在图 1 中，矩形方框是给定领域(Given Domain)，表示软件开发问题所涉及的现实世界中的要素；带有一条竖线的矩形方框是被设计的领域(Designed Domain)，表示可以由软件开发人员自由设计的描述或模型的物理实现；带有两条竖线的矩形方框是机器(Machine)领域，表示开发人员需要开发的软件解决方案；领域之间的实线连接表示共享现象(Phenomena)，描述了领域与领域之间的接口；实线的椭圆表示物理表示的描述，反映了关联的领域现象之间的关系；虚线的椭圆表示需求，需求与领域之间的虚线连接表示需求对领域现象的引用，而需求与领域之间虚线箭头表示需求对箭头指向领域的领域现象的约束. 需求对领域现象的引用和约束表达了用户期望的现实世界效果. 机器和被设计的领域共同组成了满足特定需求的解决方案. 需求引用及约束的现象构成了需求现象的集合，机器控制及引用的现象则构成了机器现象的集合. 对于科技助老系统而言，老人属于现实世界中的要素，因此把老人表示成给定领域 Older；紧急呼叫服务是开发人员要建立的软件系统，采用机器 Emergency Call Service 表示；被设计的领域 Call Record 记录老人呼叫社区服务中心的情况，如呼叫者姓名、年龄以及当前发生的应急情况等信息.

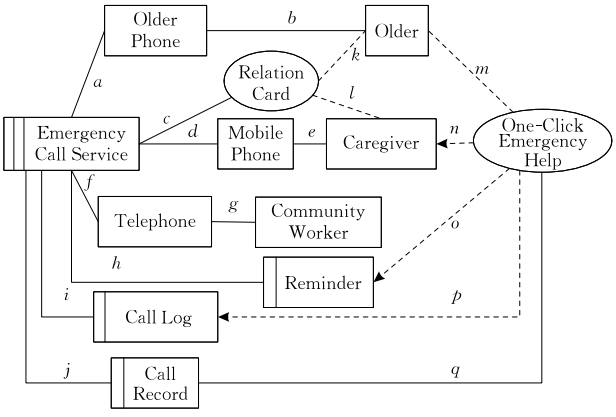

图 1　科技助老系统的初始问题图

软件开发问题框架方法总结了已有的问题分析经验，归纳了 5 种原子问题类(即问题框架)，包括命令式行为框架、需求式行为框架、信息显示框架、转换框架和工件框架. 每一种问题框架表示一类典型的软件开发问题，由问题框架图、领域特征和框架关注点三部分组成. 分析特定的软件开发问题时，可以通过匹配问题框架的方式复用已有的问题分析经验，得到给定的软件开发问题的解决方案.

软件开发问题解决方案的求解需要先后进行上下文分析、问题分析和问题渐变. 上下文分析界定了软件开发问题的边界，指明了问题所涉及的机器及现实世界中的领域要素，通过共享现象标识了机器与领域之间的连接，并使用上下文图表示问题的上下文. 问题分析将用户需求引入到上下文图之中，通过需求对现实世界中领域现象的引用及约束关系表明目标机器将要对现实世界的作用效果，建立给定问题的问题图. 问题渐变根据需求现象与机器规约现象之间的因果关系，以问题框架的框架关注点为指导，把原先使用领域现象表示的用户需求不断进行改写(Rephrase)，最终得到仅使用机器现象描述的需求，即机器规格说明. 通过问题渐变得到的机器规格说明也就是在给定的上下文环境中能够满足用户需求的解决方案. 需要指出的是，问题渐变的思想最早由 Jackson[3] 提出，但他并未给出如何实现问题渐变. 可以采用需求渐变(Requirements Progression)[8] 或问题约简(Problem Reduction)[9] 等技术实现问题渐变.

诱导出解决方案之后，可以使用正确性论证技术检验所取得的解决方案能否满足用户需求. 正确性论证[3] 是指在机器满足机器规约、环境要素满足各自的领域假设的前提下，论证给定的用户需求能否得到满足. 如果通过了正确性论证，则说明该解决方案能够满足用户需求；否则，需要重新设计给定问题的解决方案.

此外，问题框架总结了解决软件开发问题时潜在的困难，形成特定关注点. 特定关注点指明了所有软件开发问题都可能存在的求解困难，包括溢出关注点、初始化关注点、完整性关注点、身份关注点和可靠性关注点等等.

2.2　Alloy 语言

Alloy 语言[10] 是由 Jackson 提出的一种基于关

系逻辑的形式化语言，它拥有专用的 SAT 求解器 Alloy Analyzer 提供建模支持.

使用 Alloy 语言描述一个软件开发问题，需要先建立一个表示软件开发问题框架方法基本元素的元模型[11]（如图 2 所示）；然后，形式化描述所关注的软件开发问题.

```
//Constructors of the PF approach
abstract sig Domain extends Constraint{}
abstract sig Machine extends Constraint{}
abstract sig Requirement extends Constraint{}

/*Each element associates with a set of constraints which the element
   must guarantee in order to ensure the prescribed requirement */
abstract sig Constraint{}
sig Satisfied in Constraint{}
sig Unsatisfied in Constraint{}
```

图 2 软件开发问题的元模型

在图 2 中，软件问题涉及的要素被划分为给定领域、机器和需求三类，分别用抽象型构 Domain、Machine 和 Requirement 表示. 由于任何机器、现实世界领域或需求都具有各自需要满足的约束，为此引入抽象型构 Constraint 表示. 按照是否满足各自的约束，我们把软件问题中的要素（机器、现实世界领域和需求）划分为满足给定约束的元素和不满足给定约束的元素，分别用抽象型构 Satisfied 和 Unsatisfied 表示. Alloy 语言中，型构表示具有相同特征的一组元素的集合，抽象型构是一种特殊型构，其自身不能实例化，只能够通过继承（extends）关系产生其子型构，然后实例化其子型构才能产生具有特定属性的个体元素.

使用 Alloy 语言形式化描述某个软件开发问题时，先建立一个 Alloy 模型（Module）表示关注的软件开发问题；然后，定义软件开发问题中关联的现象，描述问题关联的领域、机器和需求. 以科技助老系统中给定领域“老人”的形式化描述（如图 3 所示）为例. 老人是给定领域，用型构 Older 表示. Older 是 Domain 的子型构，表示老人隶属于给定领域的范畴. 老人关联的领域现象表示成型构域，例如把老人的姓名表示为 Older 的型构域 name. 为了使得“一键拨号服务”的正确执行，需要老人施行的动作则表示成 Older 中的型构约束.

```
one sig Older extends Domain{
        name: one Name,
        ……
}{      this in Satisfied
        iff
        /*constraints to be ensured by the older
           so as to meet the requirement
        */
        ……
}
```

图 3 例示软件开发问题的形式化描述

可以使用 Alloy 语言把正确性论证表示成断言 RequirementArgument（如图 4 所示），然后利用 Alloy Analyzer 的可满足性检测功能检验断言的满足情况. 运行断言，如果 Alloy Analyzer 返回反例，表明软件开发问题的解决方案无法满足给定需求；否则，说明诱导出的解决方案能确保需求的满足. 论证某一个软件问题中需求满足情况时，可以把图 4 中的 Machine、Domain 及 Requirement 特化为给定问题中的机器集合、给定领域的集合和/或需求集合，实现面向特定问题的正确性论证.

```
assert RequirementArgument{
        Domain in Satisfied &&
        Machine in Satisfied
        implies Requirement in Satisfied
}
```

图 4 使用 Alloy 语言描述正确性论证

3 增量需求问题的定义

软件密集型系统的增量需求是指在现有需求基础上提出的新需求. 这些新需求往往与涉众需求或上下文的变化相关.

增量需求问题是一个典型的软件开发问题，用来求解增量需求的解决方案. 对于一个表示为 $W,S\vdash R$ 的软件开发问题，其增量需求问题可以以下面两种情形之一或其组合的方式出现：

模式 1. 系统将要作用的上下文环境及其约束不变，涉众在现有需求的基础上提出新需求，需要引入新的解决方案予以满足，即

$$W,S;S_{\triangle}\vdash R;R_{\triangle} \tag{2}$$

其中，$R_{\triangle}$ 表示涉众提出的新需求，$S_{\triangle}$ 表示在原有解决方案 S 基础上所开发的增量解决方案，“;”连接符最早由 Salifu 等人[12]提出，用来连接被更新要素与相应变更.

例如，在科技助老系统中，涉众提出新需求，希望系统监测居家老人的异常行为，并提供相应的应急处置服务.

模式 2. 涉众需求不变，但是软件将要作用的上下文环境发生改变，需要开发新的解决方案来适应环境变化，即

$$W;W_{\triangle},S;S_{\triangle}\vdash R \tag{3}$$

其中，$W_{\triangle}$ 表示软件系统将要作用的上下文环境所发生的变化，$S_{\triangle}$ 表示在原有解决方案 S 基础上开发的增量解决方案.

例如，科技助老系统的初始目标用户群是具有自理能力的独居老人，软件企业计划拓展该软件系统的目标用户群以服务失智失能的老年人群. 分析现有的解决方案发现现有解决方案要求老人具有自主行为的能力，而失智失能的老人可能无法满足这项假设，因此必须设计新的解决方案以服务失智失能老人.

4　增量需求解决方案求解方法

本节首先概述增量需求问题解决方案求解方法，然后描述方法中的关键步骤，包括明确增量需求问题的问题结构、抽取增量需求问题的解决方案和检验增量需求解决方案的正确性等.

4.1　方法概述

以增量需求问题的定义为基础，我们提出基于软件开发问题框架的增量需求解决方案求解方法，如图 5 所示. 本文假设现有需求已经被成功解决，在此基础上分析和求解涉众提出的增量需求问题.

(1) 分析现有需求解决方案

分析现有需求解决方案旨在抽取出现有需求及其解决方案. 其分析过程如下：

首先，明确现有需求的来源，分析现有需求问题关联的现实世界，参考文献[3]抽取出现有需求，明确其关联的软件开发问题，使用问题图描述现有需求问题的问题结构.

应用问题框架匹配[3]方法，对所取得的问题图进行问题分解、匹配问题框架和问题组合等处理，抽取出现有需求所关联的现实世界的描述，以及用来满足现有需求的软件解决方案(即现有机器的机器规约). 需要指出的是，现有需求及其关联的现实世界领域的描述，与用来满足现有需求的机器规约一起组成了现有需求问题的描述.

抽取出现有需求的解决方案之后，形式化描述现有需求问题. 按照 2.2 节软件开发问题的形式化描述过程，使用 Alloy 语言描述现有需求问题.

改写现有需求问题的上下文图，即使用添加了标签“M”的给定领域的图元表示现有需求解决方案中的机器，把现有需求解决方案中的被设计的领域直接用给定领域的图元表示. 改写后得到的上下文图就是增量需求问题关联的现实世界的一部分.

(2) 求解增量需求问题的解决方案

增量需求提出后，首要问题是明确增量需求问题的问题结构，把增量需求问题的问题结构表示成问题图.

以增量需求问题的问题图为基础，应用需求渐变[8]技术，诱导出增量需求满足时所需新机器的机器规约(即增量需求解决方案)以及增量需求问题的描述.

使用 Alloy 语言形式化描述增量需求问题. 然后，检验增量需求解决方案的正确性，即检验诱导出的解决方案能否确保增量需求的满足，同时未对原有需求的满足造成负面影响. 本文把增量需求解决方案的正确性检验问题转换成 SAT 问题(详见 4.4

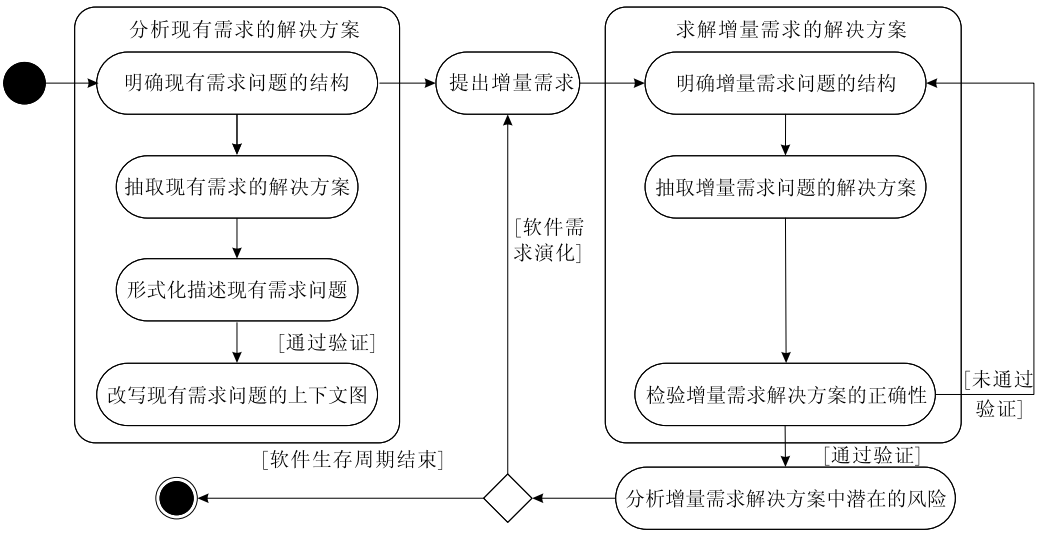

图 5　增量需求解决方案的诱导过程

节),即利用软件开发问题框架中的正确性论证技术把增量需求解决方案的检验问题表示成 Alloy 语言描述的断言,应用 Alloy Analyzer 分析增量需求解决方案的正确性.运行断言,如果 Alloy Analyzer 不返回任何反例,表明诱导出的增量需求解决方案可以满足增量需求,同时未对现有需求的满足造成任何影响.否则,说明诱导出的解决方案不能满足增量需求,或导致现有需求无法满足,需要重新求解增量需求的解决方案.重复上述过程,直至得到正确的增量需求解决方案.

(3)分析解决方案中的潜在风险

软件开发问题框架方法中特定关注点为识别增量需求解决方案中潜在的困难提供了便利.本文复用特定关注点知识,归纳了增量需求解决方案中潜在的五类风险(如表 1 所示),给出了识别并管理这些风险的策略.

表 1 增量需求解决方案的风险识别与管理

风险类别	关联的特定关注点	描述	识别方法	风险管理策略
Ⅰ	溢出关注点	领域无法及时响应由其它领域控制的共享现象	检查存在交互关系的两个领域对共享现象的触发及响应方面是否存在速率的不匹配	当无法及时响应来自其它领域的共享现象时,阻止、忽略或缓冲这些现象
Ⅱ	身份关注点	特定领域的不同实例都具有独立存在性	检查软件问题中是否需要区分隶属于同一个领域的多个实例	引入模型领域表示同一个领域的不同实例
Ⅲ	可靠性关注点	存在领域失效的风险,领域失效一旦发生将导致需求违反	分析对需求满足具有关键性影响的领域要素的可靠性	监控和诊断领域失效的发生与否,在必要时执行领域失效的修复动作
Ⅳ	初始化关注点	在软件问题中机器如何建立与现实世界领域的交互关系	分析在机器重新启动的情况下能否正确地建立机器与现实世界的交互	对机器施加额外的控制使得机器和现实世界领域的状态保持一致
Ⅴ	完整性关注点	需要保证机器、需求和现实世界领域的描述是完整的	验证在给定的环境约束下开发出的机器能否确保需求的满足	形式化论证、项目评审等等

分析增量需求解决方案中的潜在风险时,可以以表 1 作为基准,建立风险分析与管理的检查表,分析增量需求问题及其解决方案,识别出增量需求解决方案中潜在的风险要素.然后,针对每一个风险要素,在领域专家的指导下给出合适的风险管理措施.

4.2 明确增量需求问题的结构

本文采用如图 6 所示的过程确定增量需求问题的问题结构.

(1)上下文边界分析

上下文边界分析,关注软件将要作用的上下文环境的变化对增量需求问题关联的现实世界领域的影响.

由于增量需求问题和涉众提出的新需求或软件将要作用的上下文的变化相关,因此,需要确定增量需求问题匹配何种类型的增量需求问题模式.如果匹配模式 1,说明软件将要作用的上下文环境未发生变化,则直接在增量需求初始上下文图基础上进行后续的分析工作.如果匹配模式 2 或混合模式,则需要分析并记录受到上下文变化影响的领域、领域接口,然后更新现有需求的上下文图,得到增量需求问题的上下文图.

(2)系统边界分析

系统边界分析把增量需求引入到增量需求问题的上下文图中,分析需求现象,按照需求引用现象和

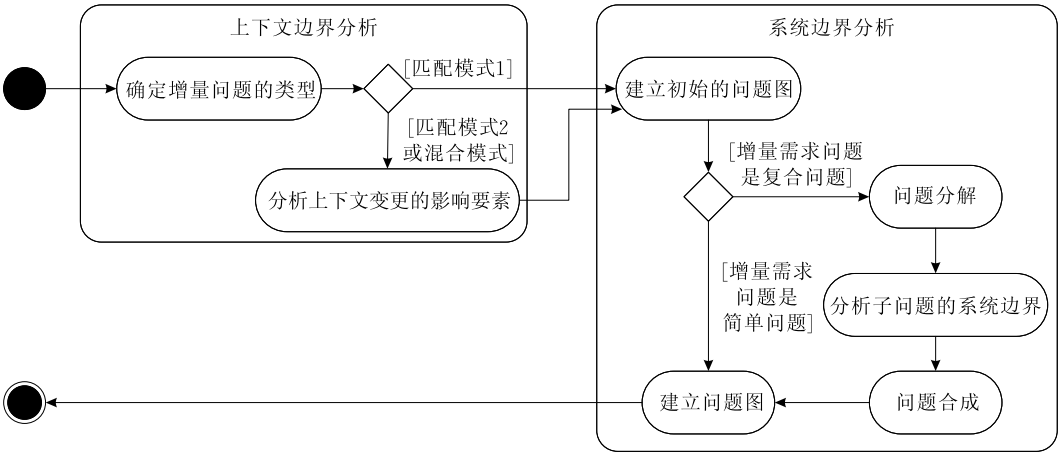

图 6 分析增量需求问题的结构

需求约束现象之间的因果关系，逐步诱导出增量需求关联的现实世界领域. 在分析需求现象的过程中，如果发现增量需求无法直接实现，还需要进行需求转换，即把原需求转换成与之等价的其它需求，分析转换后的需求所关联的需求现象，逐步抽取出与之关联的现实世界领域. 然后，引入抽象机器表示用来解决增量需求问题的计算机系统，产生增量需求问题的初始问题图.

此后，分析增量需求问题的初始问题图，即先后分析增量需求的需求现象以及抽象机器的机器规约现象，确定增量需求问题的初始问题图是简单问题还是复合问题.

分析需求现象时，如果发现需求约束的多个领域现象无法在同一个软件开发问题中实现，或被约束的领域现象之间存在多组时序关系，说明增量需求问题的初始问题图是一个复合问题. 否则，还需分析初始问题图中的机器规约现象，以进一步确定初始问题图是复合问题还是简单问题.

分析机器规约现象，即分析机器控制的现象和该机器引用的现实世界领域现象之间的因果关系. 如果发现机器规约现象中包含多组机器控制现象和现实世界领域现象之间的因果关系，则表明增量需求问题是一个复合问题. 否则，说明增量需求问题是一个简单问题.

如果发现增量需求问题是一个简单问题，则精细化描述得到的问题图，确保所有的需求现象、机器现象和领域现象都被明确的标识出来，从而明确了增量需求问题的问题结构.

但是，如果发现增量需求问题是一个复合问题，则把初始问题图分解为多个子问题，然后分析每一个子问题的问题结构. 子问题的问题结构分析过程是一个压缩的增量需求问题系统边界分析过程，需要先后进行需求解读、引入新机器、建立问题图等步骤. 重复上述过程，直至增量需求问题分解后得到的每一个子问题都仅仅匹配一个问题框架. 最后，合成所有子问题，得到增量需求问题的问题图.

4.3　求解增量需求问题的解决方案

对增量需求问题进行需求渐变[8]，得到增量需求满足时所需的新机器的机器规约以及被设计的领域的领域约束. 结合增量需求问题的问题结构，把隶属于同一个子问题的机器规约及其关联的领域约束综合起来，产生增量需求子问题的描述，并记录每一个增量需求子问题匹配的问题框架.

针对每一个增量需求子问题，查找与之匹配相同问题框架的现有需求问题的子问题；在领域专家的指导下，根据规则 1 识别并复用现有机器来解决增量需求(子)问题. 然后，把被复用的现有机器、新引入的机器、新引入的被设计的领域描述综合起来，得到增量需求问题的解决方案.

规则 1.　设 P_i 表示现有需求问题中的某个子问题，Q_j 表示增量需求问题的一个子问题. 如果：(1) P_i 和 Q_j 的拓扑结构相同，且匹配相同的问题框架；(2) P_i 与 Q_j 中机器引用现象、需求引用现象都相同；(3) P_i 与 Q_j 中机器约束现象、需求约束现象是等价的，则可以复用用来解决 P_i 的机器来解决 Q_j.

为了减少解决增量需求问题时引入新机器的数量，可以把多个新引入的机器进行组装.

规则 2.　据领域专家知识，可以把多个新引入机器进行组合，产生一个能解决多个子问题的新机器.

求解增量需求问题的过程中还应考虑替换现有机器，使得新机器可以同时满足增量(子)需求和现有(子)需求. 为此需要把新引入的机器和已实现的机器组装起来. 新引入机器与现有机器的组装过程可以参阅新机器的组装，这里不赘述.

组装新引入机器和/或现有机器时，还应侦测和处理组合关注点. 组合关注点[3,13]是指当组合多个子问题中的机器时，由于不同的子问题对同一个领域的描述存在不一致而导致的关注点. 可以参阅文献[13]识别和处理组合关注点：

(1) 如果待组合的子问题之间不存在组合关注点，则引入一个新机器来替换所有子问题中的机器，把各子问题中现实世界领域与原有机器的交互重定向到新机器上(如图 7 所示，(a)和(b)分别表示机器组合前后的情形)；

(2) 如果存在组合关注点，引入新机器协调组合关注点所涉及的机器，以解决组合关注点(如图 8

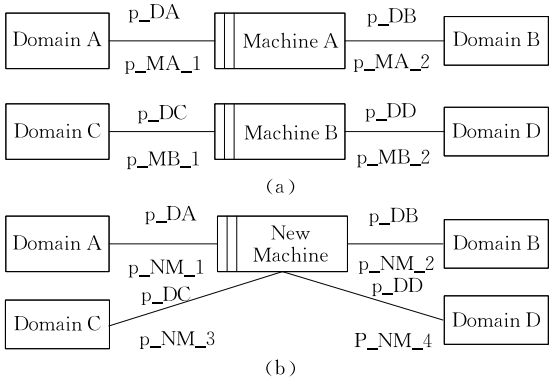

图 7　不存在组合关注点的机器组装

所示，(a)和(b)分别表示机器组装前后的情况).

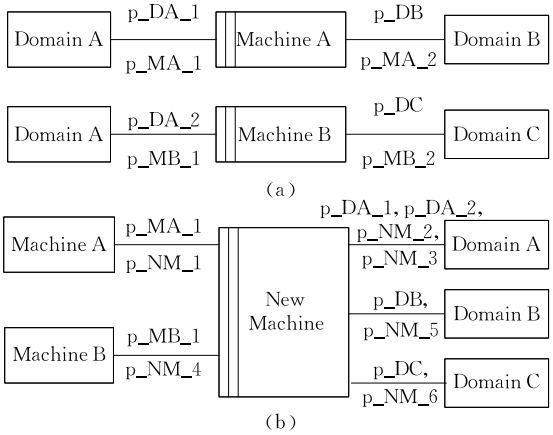

图 8 存在组合关注点的机器组装

4.4 检验增量需求解决方案的正确性

增量需求解决方案正确性的检验过程如算法 1 所示. 如果返回 true 表明增量需求解决方案是正确的，即能够满足增量需求，同时未对现有需求的满足造成任何负面影响. 如果返回 false，说明诱导出的解决方案无法满足增量需求，或导致已有需求无法满足，则重新求解增量需求的解决方案.

算法 1. 检验增量需求解决方案的正确性.

输入：Q 表示增量需求问题，

R_Q 是 Q 中必须满足的需求，

W_Q 表示 Q 中关联的现实世界领域的描述，

Q 可以由一组子问题 $\{Q_1,\cdots,Q_n\}$ 组合产生，

Q_j 是一个简单问题，

W_Q 在 Q_j 中的投影 W_{Q_j} 表示 Q_j 关联的现实世界，

R_Q 在子问题 Q_j 中的投影记作 R_{Q_j}，

M_Q 表示用来满足 R_Q 的软件规约，

M_{Q_j} 表示在 Q_j 中用来满足 R_{Q_j} 的机器，其中 $j=1,\cdots,n$，

M_P 表示现有需求问题 P 中用来满足 R_P 的机器，

W_P 是指 P 关联的现实世界描述

输出：true 或 false

```
1.  //检验增量需求子问题是否已经被解决
2.  FOR j=1 TO n
3.    形式化描述 W_Qj, M_Qj, R_Qj
4.    编写断言 Assert_j 描述 W_Qj, M_Qj ⊢ R_Qj
5.    IF 运行 Assert_j 产生反例
6.      RETURN false
7.  //检验增量需求问题是否已经被解决
8.  LET R'=R_Q1, W'=W_Q1, M'=M_Q1;
9.  FOR j=2 TO n
10.   LET R'=R'+R_Qj, W'=W'+W_Qj, M'=M'+M_Qj
11.   编写断言 Assert_j 描述 W', M' ⊢ R'
12.   IF 运行 Assert_j 导致反例产生
13.     RETURN false
14.
15. //检验已有需求和增量需求能否都得到满足
16. LET R'=R'+R_P, W'=W'+W_P, M'=M'+M_P
17. 编写断言 Assert 描述 W', M' ⊢ R'
18. IF 运行 Assert 导致反例产生
19.   RETURN false
20. RETURN true
```

(1) 检验增量需求子问题解决情况

由于确定增量需求问题结构时增量需求问题 Q 被分解为 n 个子问题($n\geq 1$)，记作 $\{Q_1,\cdots,Q_n\}$，增量需求 R_Q 在子问题 Q_j 中的投影记作 R_{Q_j}，Q 关联的现实世界 W_Q 在 Q_j 中被投影为 W_{Q_j}，M_{Q_j} 表示在给定的环境约束 W_{Q_j} 下用来满足 R_{Q_j} 的机器规约. 检验增量需求子问题解决情况，即检验在给定的环境约束 W_{Q_j} 下机器规约 M_{Q_j} 能否确保 R_{Q_j} 的满足.

检验增量需求子问题解决情况时，针对每一个增量需求子问题 Q_j：

① 首先，使用 Alloy 描述 Q_j，即把 Q_j 中机器、需求和给定领域的形式化描述封装在一个 Alloy 模块中.

② 然后，定义断言 Assert_{Q_j} 描述 Q_j 关联的正确性论证，并以 Q_j 中机器、需求和给定领域要素的个数之和作为状态空间的搜索深度. 以科技助老系统增量需求问题 Q 为例. 假设科技助老系统增量需求“老人紧急情况自动侦测与处置”问题中，子问题 Q_1 中现实世界领域包括社区工作人员 Community Worker，远红外传感器 Infrared Sensor、老人 Older 和被设计的领域“老人与远红外传感器之间的映射”(记作 Older～Sensor Mapping)，机器 Older～Sensor Mapping Controller 表示用来满足子需求“建立老人与远红外传感器之间的映射关系(Older～Sensor Mapping Definition，记作 OlderSensorMappingReq)”. 为 Q_1 定义断言 OlderSensorMappingArgument(如图 9 所示)检验子需求的满足情况. 由于 Q_1 关联的软件开发问题要素的个数之和为 6，因此设置断言的搜索深度设置为 6，运行此断言即可检验 Q_1 的解决情况.

```
assert OlderSensorMappingArgument{
        (CommunityWorder+Older+InfraredSensor+
        OlderSensorMapping) in Satisfied &&
        OlderSensorMappingController in Satisfied
        implies OlderSensorMappingReq in Satisfied
}

check OlderSensorMappingArgument for 6
```

图 9 断言 OlderSensorMappingArgument

③运行断言 $Assert_{Q_j}$，查看 Alloy Analyzer 的返回结果. 如果返回反例，表明 Q_j 中机器规约无法满足 R_{Q_j}，需要重新求解 Q_j 的解决方案. 如果无反例产生，则重复上述过程检验 Q 的其它子问题中需求的满足情况.

(2) 检验增量需求问题的满足情况

为 Q 新建一个 Alloy 模块，表示 Q 的形式化描述，记作 $module_Q$. 然后，采用增量方式合成(Compose)增量需求问题的子问题，产生组合问题，使用组合问题中的领域现象改写已被组合的所有子问题中的需求、机器和需求，生成组合问题的形式化描述. 把图 4 所示的断言 RequirementArgument 引入到组合问题的形式化描述中，设置运行断言的搜索深度为组合问题中的机器、给定领域和需求等要素的数量之和. 运行断言，查看是否有反例. 如果无反例返回，说明已组合的各子问题都已被正确解决，则继续上述过程，直至组合产生增量需求问题并确认增量需求已经被正确解决为止. 但是，一旦发现反例，表明存在组合关注点，需要引入新的机器来识别并处理子问题组合时的组合关注点，即需要重新求解增量需求问题的解决方案.

(3) 检验增量需求解决方案对现有需求的影响

合成现有需求问题和增量需求问题，形式化描述合成得到的组合问题，产生 Alloy 模块 $module_{P\&Q}$. 把 RequirementArgument 引入到该模块之中，设置运行断言所需的状态空间深度为组合问题中的机器、需求和给定领域的数量之和，运行断言，查看是否返回反例. 如果返回反例，说明诱导出的增量需求解决方案导致现有需求无法满足，则需要重新求解增量需求解决方案. 如果无反例，说明增量需求问题已被正确解决，同时并未影响现有需求的满足.

5　案例分析

本节以科技助老系统增量需求问题的求解阐明如何应用本文方法，同时验证方法的有效性.

5.1　分析现有需求及其解决方案

5.1.1　抽取现有需求及其解决方案

设 R_1 表示现有需求一键求助“One-Click Service”，问题 P_1 表示求解 R_1 解决方案的软件开发问题. 假设 R_1 已经被正确实现.

首先，通过阅读需求规格说明、观察现有系统的行为等方式，抽取出 P_1 的问题结构，建立如图 1 所示的问题图. 然后，应用问题框架匹配技术把 P_1 分解为 4 个子问题，分别用 $P_{1.1}$、$P_{1.2}$、$P_{1.3}$ 和 $P_{1.4}$ 表示(如图 10 所示). $P_{1.1}$ 和 $P_{1.3}$ 匹配信息显示框架，$P_{1.2}$ 和 $P_{1.4}$ 匹配命令式行为框架. 在匹配问题框架的过程中，R_1 投影为 4 个子需求，分别用 $R_{1.1}$“Relation Card Management”、$R_{1.2}$“One-Click Dial-up”、$R_{1.3}$“Call Record Generation”和 $R_{1.4}$“Emergency Disposal”表示. 再根据问题框架的框架关注点，分析每一个子问题中领域现象之间的因果关系，得到子问题中的机器规约和领域约束. 所有子问题中的机器规约(如表 2 所示)与被设计的领域共同组成了 R_1 的解决方案.

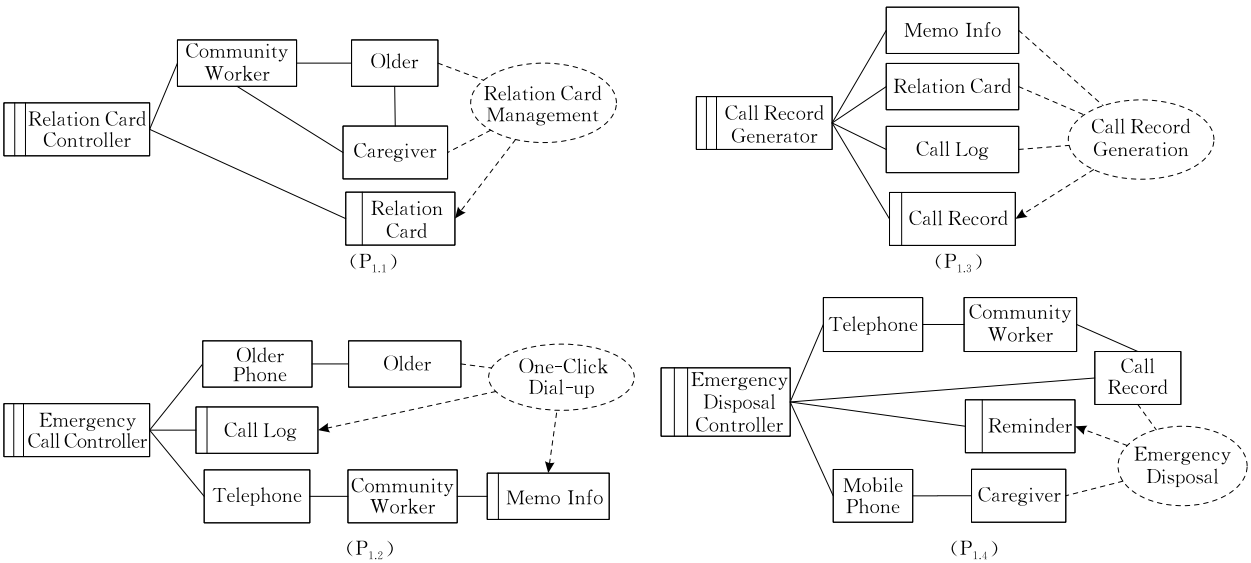

图 10　P_1 的问题分解

表 2 R_1解决方案中的机器规约

机器名称	机器规约
Relation Card Controller	在联系卡中记录老人以及老人关爱员的姓名、电话号码等信息
Emergency Call Controller	把由老人手机发出的语音信号转发给社区服务中心的热线电话，并且在呼叫日志 Call Log 中记录呼叫端的电话号码
Call Record Controller	读取呼叫记录中的拨号记录，查询联系卡确定呼叫者的姓名. 然后由社区中心的工作人员根据呼叫者报告的情况产生呼叫记录 Call Record
Emergency Disposal Controller	把来自社区电话 Telephone 的语音信号转发给关爱员的手机 Mobile Phone 上，向关爱员给出提醒

5.1.2 形式化描述现有需求问题

引入图 2 所示的元模型表示软件开发问题框架的基本元素，使用 Alloy 语言描述现有需求问题的子问题，再通过子问题组合产生现有需求问题的形式化描述.

(1) 描述现有需求子问题

为 $P_{1.1}$新建一个 Alloy 模块 $P1.1$，封装 $P_{1.1}$的形式化描述. 在 $P1.1$ 中，首先定义领域现象电话号码 PhoneNumber 和姓名 Name；然后，形式化描述 $P_{1.1}$中的机器 RelationCardController、现实世界领域(包括老人 Older、老人关爱员 Caregiver 和老人联系卡 RelationCard)和需求“老人联系卡的建立”(记作 OlderRegistrationReq)，如图 11 所示. 然后采用上述方法，形式化描述 P_1的其它子问题.

```
//phenomena
sig Name{}
abstract sig PhoneNumber{}

/* descriptions of domains, such as Older, Caregiver etc. */
one sig Older extends Domain{
    name: one Name,
    phoneNumber: one OlderPhoneNumber
}{ this in Satisfied }
……

//specification of machine OlderRegistrationController
one sig OlderRegistrationController extends Machine{
    older: Older,
    caregiver: Caregiver,
    relationCard: one RelationCard
}{ this in Satisfied iff
    relationCard.oName=older.name &&
    relationCard.cName=caregiver.name &&
    relationCard.oPhoneNumber=older.phoneNumber &&
    relationCard.cPhoneNumber=caregiver.phoneNumber
}

//requirement descriptions
one sig OlderRegistrationReq extends Requirement{
    older: Older,
    caregiver: Caregiver,
    relationCard: RelationCard
}{ this in Satisfied iff
    relationCard.oName=older.name &&
    relationCard.cName=caregiver.name &&
    relationCard.oPhoneNumber=older.phoneNumber &&
    relationCard.cPhoneNumber=caregiver.phoneNumber
}}
```

图 11 例示现有需求子问题的形式化描述

(2) 组合子问题，产生现有需求问题的描述

本文采用增量方式组合现有需求问题的子问题，最终得到现有需求问题的形式化描述. 以子问题 $P_{1.1}$和 $P_{1.2}$组合过程为例：

① 首先，建立一个 Alloy 模型(Module) $P1$ 表示现有需求问题，复制 $P_{1.1}$的形式化描述到 $P1$ 之中，即集成子问题 $P_{1.1}$；

② 在 $P1$ 中添加 $P_{1.2}$的形式化描述，根据 Alloy Analyzer 的语法检测功能识别并处理这两个子问题中命名重复的领域现象、领域和/或机器，把两个子问题中都包含的领域或机器的描述组合起来. 例如，在 $P_{1.1}$和 $P_{1.2}$中都包含对领域 Older 的描述，由于不存在组合关注点，只需消除重复定义的领域现象，组合这两个子问题中 Older 的领域约束即可得到子问题组合后 Older 的形式化描述(如图 12 所示). 按照同样的方式组合 $P_{1.1}$和 $P_{1.2}$中其它要素的描述.

采用同样的方式，先后组合 P_1的其它子问题，即把 $P_{1.3}$和 $P_{1.4}$的形式化描述合成到 P_1之中，最终得到现有需求问题 P_1的形式化描述.

5.1.3 改写现有问题的上下文图

改写 P_1的上下文图，即把已实现的机器看作给定领域，使用标记“M”标识，得到如图 13 所示的上下文图.

5.2 求解增量需求的解决方案

涉众拟在软件的下一个发布版本中增加“老人紧急情况自动侦测与处置”服务. 令 R_2表示增量需求“老人紧急情况自动侦测与处置”，P_2表示 R_2的求解问题. R_2的描述如表 3 所示.

表 3 增量需求 R_2的描述

需求编号	R_2
需求名称	老人紧急情况自动侦测与处置
需求描述	需要开发一个软件系统，自动侦测老人在居所内的日常行为. 一旦发现老人行为异常，则自动向老人关爱员发送提醒短信.

```
one sig Older extends Domain{
    //phenomena of Older within P1.1
    name: lone Name,
    phoneNumber: lone PhoneNumber,
    caretakerName: lone Name,
    communityWorker: CommunityWorker,//referred domain
    caregiver: Caregiver,

    //phenomena of Older within in P1.2
    --naming conflict results in rephrased descriptions
    //name: lone Name, --remove the duplicate named element
    senderConversation: EmergencyCallCallerConversation,
    emergency: Emergency,
    phone: OlderPhone, --relevant domain
    --remove the duplicate named element
    //phoneNumber: lone PhoneNumber
}{
    this in Satisfied
    iff --composed domain constraints in P1.1 and P1.2
    //domain constraints of Older within in P1.1
    communityWorkder.relationCardInfo.oName
        =name &&
    communityWorkder.relationCardInfo.oPhoneNumber
        =phoneNumber &&
    caretakerName=caregiver.name

    //domain constraints of Older within in P1.2
    senderConversation.oName=name &&
    senderConversation.emergency=emergency &&
    phone.callerVoiceCurrentOfEmergencyCall.conversation
        =senderConversation &&
    phoneNumber=phone.phoneNumber
}
```

图 12　例示组合多个子问题中的领域描述

P_2不涉及软件将要作用的上下文环境变化，因此匹配增量需求问题模式 1，现有需求的上下文就是 P_2关联的现实世界的一部分.

5.2.1　确定增量需求问题的结构

(1) 建立初始问题图

为了确定增量需求问题的问题结构，首先根据 R_2的需求引用和需求约束，对现有需求的上下文图进行投影，发现 R_2引用 Older、Caregiver 的领域现象，R_2的需求约束涉及预警器 Alarm、短信提醒 Reminder SMS. 然后，引入抽象机器 Abnormal Behavior Disposal Service 表示将要开发的 R_2的解决方案，得到 P_2的初始问题图(如图 14 所示).

(2) 抽取增量需求问题的结构

分析 P_2的初始问题图发现 R_2的需求现象涉及多组需求引用现象与需求约束现象之间的因果关系. 由此可以确定 P_2的初始问题图是一个复合问题，需要对 P_2进行问题分解与合成，才能确定 P_2的系统边界. 根据 R_2的自然语言描述，P_2被分解为 $P_{2.1}$和 $P_{2.2}$两个子问题. $R_{2.1}$(如表 4 所示)和 $R_{2.2}$(如表 5 所示)分别表示 R_2在 $P_{2.1}$和 $P_{2.2}$中的投影.

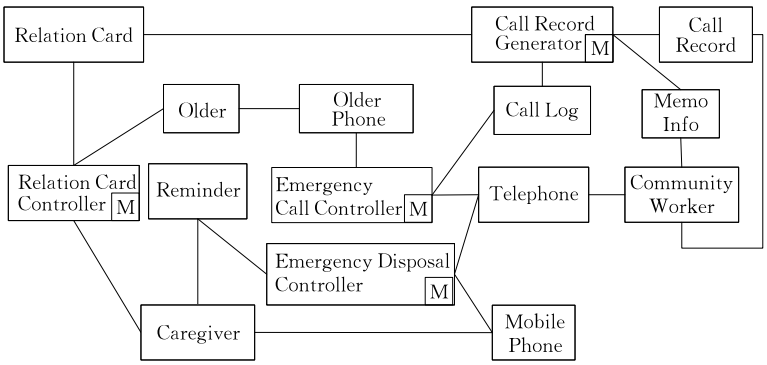

图 13　增量需求问题关联的现实世界

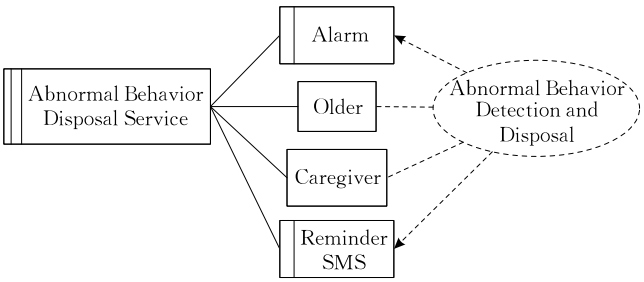

图 14　P_2的初始问题图

表 4　子需求 $R_{2.1}$的描述

需求编号	$R_{2.1}$
需求名称	老人行为异常自动侦测
需求描述	需要开发一个软件系统能够自动侦测老人在居所内的日常行为. 一旦发现老人行为异常，报警器将响起，并发出求救信号.

表 5　子需求 $R_{2.2}$的描述

需求编号	$R_{2.2}$
需求名称	老人行为异常的处置
需求描述	需要开发一个软件系统接收来自预警控制器的求救信号，根据求救信号识别出信号是与哪位老人关联的传感器触发的，然后向相应的老人关爱员发送提醒短信.

在 $P_{2.1}$中，老人 Older 是可叫牌领域，具有自主行为的能力，即老人的行为是不可预测的. 需要把“老人行为的监控问题”转换成“对老人所在地点的监控问题”，即把需求 $R_{2.1}$转换成需求 $R_{2.1'}$(如表 6 所示). 根据 $R_{2.1'}$的需求现象，引入安装在客厅的远红外传感器 Infrared Sensor 自动侦测老人在客厅

的活动情况，使用被设计的领域求救信号 Distress Signal 作为 $R_{2.1'}$需求约束的领域，然后引入新机器形成如图 15 所示的问题图.

表 6 把 $R_{2.1}$特化为 $R_{2.1'}$

需求编号	$R_{2.1'}$
需求名称	老人行为异常自动侦测
需求描述	需要开发一个软件系统自动侦测老人在客厅的活动情况.如果在设定的时间段内(如 10 h 内)未侦测到老人的任何活动迹象，报警器将响起，并发出求救信号.

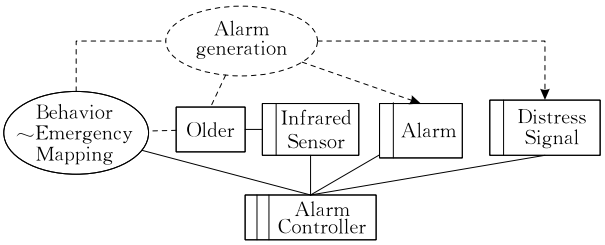

图 15 $P_{2.1}$的问题图

子问题 $P_{2.2}$涉及的领域包括 Caregiver、Older、Infrared Sensor、SMS Reminder 和 Distress Signal. 引入机器 Abnormal Behavior Controller，建立 $P_{2.2}$的初始问题图(如图 16 所示).

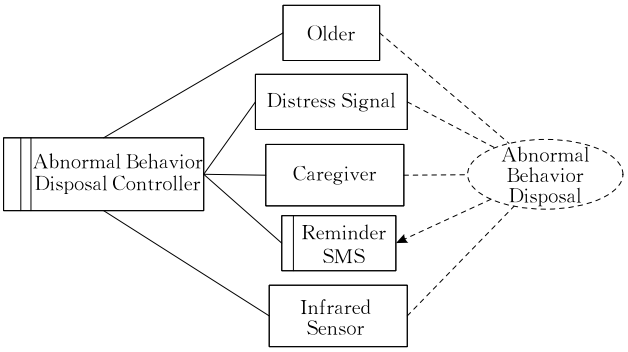

图 16 $P_{2.2}$的初始问题图

分析图 16 中的需求现象，发现 $P_{2.2}$涉及 3 组领域现象之间的因果关系，即老人与关爱员之间的映射、远红外传感器与被监控的老人之间的映射，以及根据远红外传感器触发的求救信号向老人关爱员发送求助短信，说明 $P_{2.2}$的初始问题图是一个复合问题.按照需求现象之间的因果关系，把 $P_{2.2}$分解为 3 个子问题 $P_{2.2.1}$、$P_{2.2.2}$和 $P_{2.2.3}$(如图 17 所示).

由于 P_2的所有子问题的问题结构均已明确，因此合成子问题 $P_{2.1}$、$P_{2.2.1}$、$P_{2.2.2}$和 $P_{2.2.3}$得到 P_2的问题图(如图 18 所示).

5.2.2 增量需求解决方案的求解

本文采用需求渐变技术求解增量需求满足时所

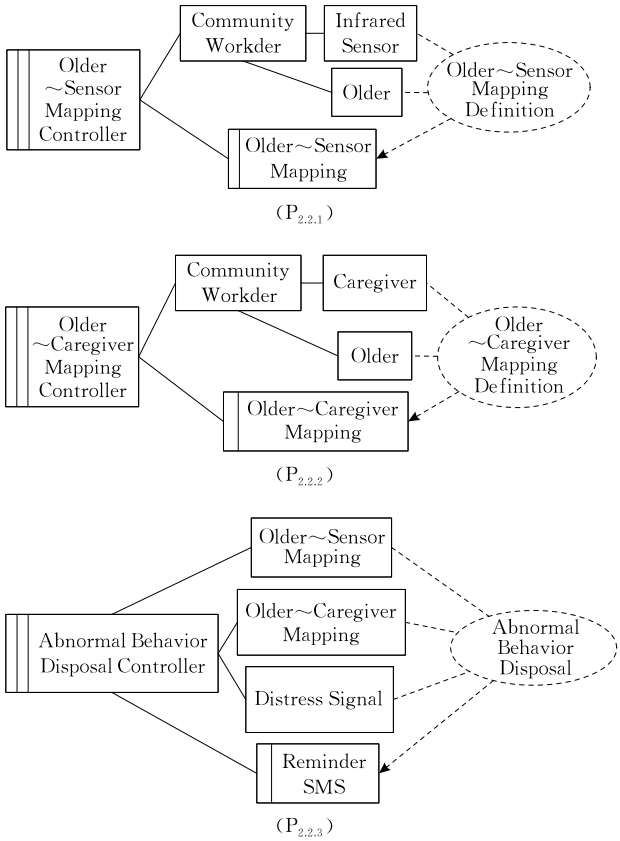

图 17 $P_{2.2}$的子问题

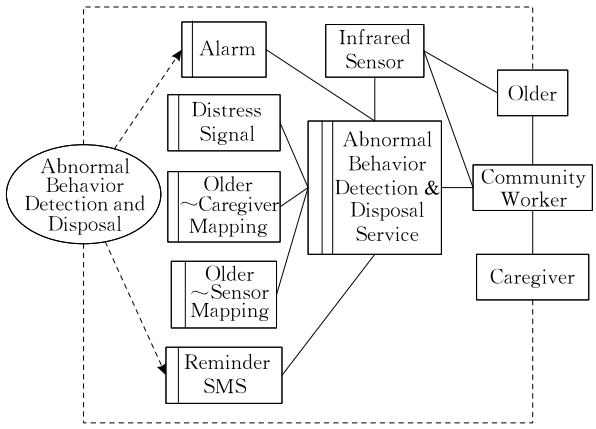

图 18 P_2的问题图

需的机器规约和被设计的领域的领域约束.

(1)问题分解

把 P_2分解为 4 个问题，即 $P_{2.1}$、$P_{2.2.1}$、$P_{2.2.2}$和 $P_{2.2.3}$.

(2)诱导机器规约及其关联的领域约束

利用框架关注点分析技术，为每一个子问题抽取出需求满足时所需的领域假设和机器规约.然后，合并隶属于同一个机器或领域的约束，得到新引入机器的机器规约及其关联的领域约束.以 $P_{2.1}$的需

求渐变为例($P_{2.1}$的需求渐变过程如图 19 所示，需求渐变得到的领域约束和机器规约如表 7 所示)：

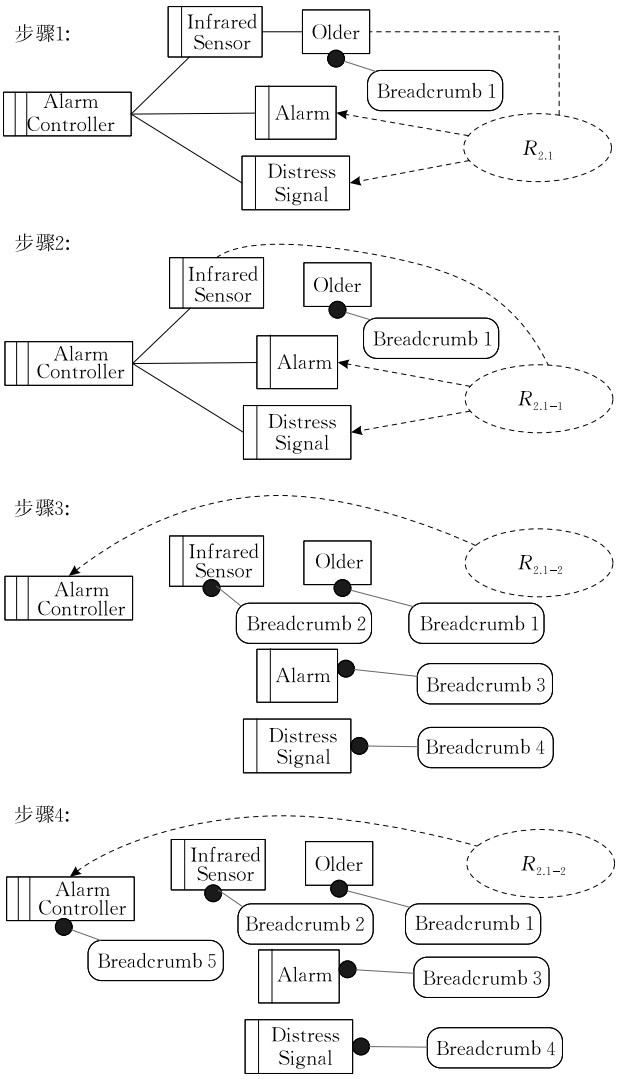

图 19　子问题 $P_{2.1}$的需求渐变过程

表 7　$P_{2.1}$的面包屑

编号	面包屑的含义
Breadcrumb 1	老人在过去的 12h 内从未在客厅活动
Breadcrumb 2	远红外传感器在过去 12h 内未侦测到任何活动迹象，则触发老人活动异常脉冲
Breadcrumb 3	报警器响起
Breadcrumb 4	求救信号记录了是由哪个远红外传感器引发的老人动作异常预警信号
Breadcrumb 5	根据接收到的老人活动异常脉冲，产生求救信号，并向报警器发送启动信号

① 首先，依据 $R_{2.1}$的需求描述，抽取出 Older 的领域假设，用面包屑[①] Breadcrumb 1 表示；

② 在 Older 满足 Breadcrumb 1 描述的领域约束的情况下，使用 Infrared Sensor、Alarm 和 Distress Signal 的领域现象改写 $R_{2.1}$，把改写后的需求用 $R_{2.1\text{-}1}$表示.

③ 由于改写后的需求与 Older 的现象无关，则把改写后的需求推向 Infrared Sensor，得到步骤 2 所示的结果.

④ 按照同样的方式，渐进地抽取出面包屑、改写需求，最终得到 Alarm Controller 的机器规约，用 Breadcrumb 5 表示.

比对增量需求子问题和现有需求子问题，发现子问题 $P_{1.1}$和 $P_{2.2.2}$符合规则 1 的描述，即 $P_{1.1}$和 $P_{2.2.2}$是等价的. 因此复用 $P_{1.1}$中机器 Relation Card Controller 和被设计的领域 Relation Card 来管理老人和老人关爱员之间的监护关系. 复用已实现的机器及被设计的领域要求对增量需求问题进行适当的改写，因此，修改子问题 $P_{2.2.3}$，即把 $P_{2.2.3}$中的 Older～Caregiver Mapping 更新为已实现的被设计的领域 Relation Card 产生子问题 $P_{2.2.3'}$(如图 20 所示).

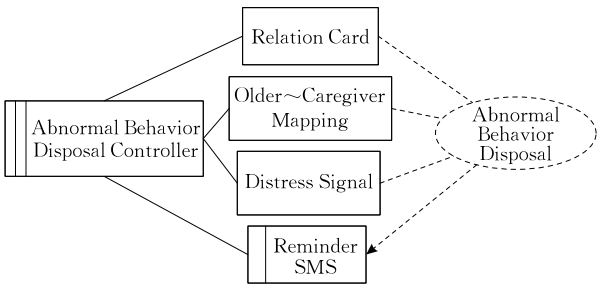

图 20　子问题 $P_{2.2.3'}$

然后，把所有子问题中的机器规约、被设计的领域的描述综合起来形成增量需求的解决方案.

5.2.3　检验增量需求解决方案的正确性

按照算法 1 检验增量需求解决方案的正确性：

(1)使用 Alloy 形式化描述每一个增量需求子问题，分别为 $P_{2.1}$、$P_{2.2.1}$、$P_{2.2.2}$、$P_{2.2.3'}$编写断言验证子问题中的机器能否满足子问题中的需求，发现诱导出的机器规约能够满足子需求.

(2)建立增量需求问题的 Alloy 模型 $P2$，采用增量方式把增量需求子问题的描述集成到 $P2$ 中，使用断言 RequirementArgument 检验子问题集成过程中由子问题组合而产生的组合需求的满足情况. 产生增量需求问题描述的过程中，运行断言，发现无反例，说明增量需求的解决方案能够满足增量需求.

(3)把现有需求问题的描述集成到 $P2$ 中，根据 $P2$ 中当前拥有的给定领域、机器和需求的数量之和

① 面包屑表示需求渐变过程中得到的领域约束或机器规约的片段.

调整 RequirementArgument 的搜索深度，检验增量需求解决方案是否影响现有需求的满足，运行断言发现无反例，说明增量需求解决方案对现有需求的满足无负面影响.

(4)把更新后的 $P2$ 作为新的现有需求问题的描述，为求解新的增量需求问题做准备.

5.3 分析潜在的风险

得到增量需求解决方案后，我们复用特定关注点的知识，发现 R_2 的解决方案存在如表 8 所示的潜在风险.

表 8 增量需求解决方案潜在的风险

风险编号	复用的特定关注点	潜在的风险
1	溢出关注点	老人的行为是自主的且不可预测的，存在老人动作过快而远红外传感器无法及时响应的风险
2	可靠性关注点	增量需求的满足要求远红外传感器具有高可靠性，因此需要侦测和处置远红外传感器的失效
3	身份关注点	老人的宠物有可能触发远红外传感器的动作，因此需要区分远红外传感器感应到的动作是老人的行为还是宠物的行为，以避免误判

识别增量需求解决方案中潜在的风险之后，可以复用特定关注点的处置技术完善增量需求的解决方案，为后续的设计决策提供支持. 例如，可以使用“忽略”策略处置风险 1，即远红外传感器忽略其无法及时响应的老人动作；而风险 3 要求设计具有更强领域约束的远红外传感器，即远红外传感器具有形状区分功能，能够区分人的动作和宠物的动作. 一旦涉众决定处置某个风险要素，将产生新的增量需求问题.

6 相关工作

技术更新、软件运行环境变化以及业务目标的改变共同驱动需求变更，导致需求演化问题的产生. 需求演化[14]是指，对软件需求及其制品进行持续更新来满足日益频繁的需求变更，并对受到需求变更影响的软件制品进行同步更新. 应对需求演化的方法[15]包括软件维护，或通过产品线工程、构造自适应系统等方式重构软件系统.

本文关注由增量需求的提出而导致的需求演化问题，提出一个基于软件开发问题框架的增量需求解决方案求解方法(解决方案表示成需求规格说明)，为通过软件维护实现增量需求解决方案提供支持. Ernst 等人[5]从用户需求、软件解决方案和给定的领域三者之间的因果关系出发，提出了面向目标的增量需求解决方案的诱导方法. 在该方法中，用户需求使用目标模型中的目标表示，现有解决方案中软件系统的行为用任务(Task)表示，与用户需求关联的领域假设通过开发人员所拥有的领域知识展示. 增量需求解决方案的求解过程如下：首先在领域知识指导下进行目标操作化(Operationalization)[16]，得到必须的新任务(目标操作化的过程可以看作是一种典型的软件维护活动)并更新任务知识库；形式化描述增量需求，然后利用预定义的一组规则检索知识库，检索得到一个候选任务集来满足增量需求. 如果多个候选任务集可以满足增量需求，则根据涉众偏好选择一个最佳方案. 与 Ernst 的方法相比，本文把现有解决方案中的机器规约作为可复用的需求制品，应用基于软件开发问题框架的需求分析方法求解增量需求解决方案. 为了显式的表示需求变更，Brier 等人关注需求变更和/或上下文变化对现有需求的影响，提出变更问题图(Change Problem Diagram)[17]的概念. 变更问题图采用图示的方式把受到需求变更、上下文变化影响的要素在问题图中标识出来，但并未涉及增量需求解决方案的求解.

在产品线领域，Tun 等人[6]提出了一个基于软件开发问题框架的需求演化问题的求解方法. 针对演化需求(即特征)，该方法把现有特征的解决方案(即机器规约)作为增量特征问题所关联的现实世界的一部分，把为了解决增量特征问题而引入的新机器作为对现实世界领域和已实现的机器的包装器，在明确增量需求问题的基础上使用问题渐变得到新机器的规约，再利用正确性论证对增量特征的满足情况进行形式化和/或非形式化的验证. Tun 的方法与本文工作都把已实现的机器规约看作黑箱并作为可复用的基本单元，通过引入新机器来改变现有需求的问题结构以此来满足增量需求的需要，都应用正确性论证技术验证解决方案的有效性. 与 Tun 的方法相比，本文的方法关注增量需求问题，强调从增量需求问题发生的根本原因出发，分析增量需求问题的问题结构，采用增量、迭代的方式诱导出增量需求的解决方案，把增量需求解决方案的检验问题转换成可满足问题求解，还复用特定关注点评价增量需求解决方案中的潜在风险. 在增量需求解决方案的求解过程中，本文首先引入抽象机器表示用来解决增量需求问题的机器，利用需求现象和机器现象的解读，渐进地澄清增量需求问题的问题结构，然

后利用需求渐变诱导出增量需求解决方案.我们在前期工作[18]中关注软件产品线中由于涉众偏好变更、系统上下文改变而导致的需求演化问题.基于软件开发问题框架的需求分析方法,我们分析并抽取出需求演化模式.以预定义的需求演化模式为基础,对需求演化问题进行问题分析与组合来确定受到需求演化影响的特征集合,更新原有的特征模型,结合用户偏好及特征的挣值(Earned Value)对最新更新的特征模型进行特征绑定,得到能够解决需求演化问题的特征配置.与前期工作相比,本文关注增量需求问题,提出基于软件开发问题框架的增量需求解决方案求解方法,采用形式化方法检验解决方案的正确性,还复用特定关注点评估解决方案中潜在风险.

重构业务系统使之具备自适应能力是应对频繁的需求变更、有效处理需求演化问题的重要手段[15,19].Wang 等人[20]和 Elkhodary[21]等人在业务系统基础上,开发自适应系统构件,使得自适应系统构件和业务系统组成闭环控制环路,利用运行时软件重配置处置运行时软件失效.为了应对软件运行环境变更,Ali 等人[22]提出上下文化的目标模型(Contextual Goal Model)把软件运行时上下文要素和涉众目标关联起来,通过运行时目标模型动态绑定指导软件重配置,来确保涉众需求的持续满足.Salifu 等人[23]以领域专家知识为指导,分析软件运行时的上下文可变性,定义上下文感知的自适应规则并开发自适应系统构件,根据预定义的自适应规则进行上文感知的自适应动作,以响应软件运行时的上下文变化.

7　总结与展望

软件密集型系统的新需求通常以增量、迭代的方式提出并解决,需要在复用现有方案的基础上抽取出增量需求的解决方案,并尽早检验和评估诱导出的解决方案.为此,本文提出了一种基于软件开发问题框架的软件密集型系统增量需求问题的求解方法.本文的主要工作包括:(1)基于软件开发问题框架的需求分析技术,采用一个增量、迭代的求解过程,诱导出增量需求的解决方案;(2)利用 SAT 求解器检验增量需求解决方案的正确性,为解决方案的评价提供即时反馈;(3)复用软件开发问题的求解经验,评估增量需求解决方案的潜在风险,为优化解决方案提供支持.

该方法的主要特点有:(1)在有效利用已有软硬件资产的基础上求解增量需求问题;(2)在需求工程过程中检验增量需求解决方案的正确性.与软件生存周期后期进行方案验证相比,可有效减少软硬件投资方面的浪费.

后续工作将从方法改进、工具开发和应用实践三方面展开.目前,本文诱导出的增量需求的解决方案是采用机器现象描述的机器规约,解决方案的描述粒度有待进一步细化.今后,将尝试引入状态机模型描述的机器规约,以提供更精细粒度的解决方案;将变更问题图与本文方法结合起来,显式地分析和表示增量需求对现有需求问题的影响.开发基于本文方法的需求建模工具,来帮助软件开发人员分析和求解增量需求问题,也是未来工作的重要内容.实践方面,将致力于选择一些典型的软件密集型系统开展更大规模的案例研究,进一步检验方法的有效性.

参　考　文　献

[1] Pohl K. Requirements Engineering: Fundamentals, Principles, and Techniques. Beijing: China Mechine Press, 2012 (in Chinese)
(Pohl K. 需求工程:基础、原理和技术. 北京:机械工业出版社, 2012)

[2] Neill C J, Laplante P A. Requirements engineering: The state of the practice. IEEE Software, 2003, 20(6): 40-45

[3] Jackson M. Problem Frames: Analysing and Structuring Software Development Problems. Beijing: China Mechine Press, 2005(in Chinese)
(Jackson M. 软件开发问题框架:现实世界问题的结构化分析. 北京:机械工业出版社, 2005)

[4] Dao T, Kang K. Mapping features to reusable components: A problem frames-based approach//Proceedings of the 14th International Software Product Line Conference (SPLC 2010). Jeju Island, South Korea, 2010: 377-392

[5] Ernst N, Borgida A, Jureta I. Finding incremental solutions for evolving requirements//Proceedings of the 19th IEEE International Requirements Engineering Conference (RE 2011). Trento, Italy, 2011: 15-24

[6] Tun T, Trew T, Jackson M. Specifying features of an evolving software system. Software: Practice and Experience, 2009, 39(11): 973-1002

[7] Biere A, Cimatti A, Clarke E M, Zhu Y. Symbolic model checking without BDDs//Proceedings of the 5th Interantional Conference on Tools and Algorithms for the Construction and Analysis of Systems(TACAS'99). Amsterdam, Netherlands, 1999: 193-207

[8] Seater R, Jackson D, Gheyi R. Requirement progression in problem frames: Deriving specifications from requirements. Requirements Engineering, 2007, 12(2): 77-102

[9] Rapanotti P, Hall J, Li Z. Deriving specifications from requirements through problem reduction. IEE Proceedings-Software, 2006, 153(5): 183-198

[10] Jackson D. Software Abstractions: Logic, Language and Analysis. Cambridge, Massachusetts: MIT Press, 2012

[11] Kang E. A framework for dependability analysis of software systems with trusted bases [M. S. dissertation]. Massachusetts Institute of Technology, Cambridge, Massachusetts, 2010

[12] Salifu M, Yu Y, Nuseibeh B. Specifying monitoring and switching problems in context//Proceedings of the 15th IEEE International Requirements Engineering Conference (RE 2007). New Delhi, India, 2007: 211-220

[13] Laney R, Barroca L, Jackson M, Nuseibeh B. Composing requirements using problem frames//Proceedings of the 12th IEEE International Conference on Requirements Engineering (RE 2004). Kyoto, Japan, 2004: 122-131

[14] Harker S D P, Eason K D, Dobson J. The change and evolution of requirements as a challenge to the practice of software engineering//Proceedings of the IEEE International Symposium on Requirements Engineering (RE'93). San Diego, USA, 1993: 266-272

[15] Ernst N, Mylopoulos J, Wang Y. Requirements evolution and what (research) to do about it//Proceedings of the Design Requirements Workshop. Cleveland, USA, 2007: 186-214

[16] van Lamsweerde A. Goal-oriented requirements engineering: A guided tour//Proceedings of the 5th IEEE International Symposium on Requirements Engineering (RE 2001) Toronto, Canada, 2001: 249-262

[17] Brier J, Rapanotti L, Hall J. Capturing change in socio-technical system with problem frames//Proceedings of the 11th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ 2005). Porto, Portugal, 2005

[18] Peng X, Yu Y, Zhao W. Analyzing evolution of variability in a software product line: From contexts and requirements to features. Information & Software Technology, 2011, 53(7): 707-721

[19] Jeff K, Jeff M. Self-managed systems: An architectural challenge//Proceedings of the Workshop on the Future of Software Engineering. Minneapolis, USA, 2007: 259-268

[20] Wang Y, McIlraith S A, Yu Y, Mylopoulos J. An automated approach to monitoring and diagnosing requirements//Proceedings of the 22nd IEEE/ACM International Conference on Automated Software Engineering (ASE 2007). Atlanta, USA, 2010: 293-302

[21] Elkhodary A M, Esfahani N, Malek S. FUSION: A framework for engineering self-tuning self-adaptive software systems//Proceedings of the 18th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE 2010). Santa Fe, USA, 2010: 7-16

[22] Ali R, Dalpiaz F, Giorgini P. A goal-based framework for contextual requirements modeling and analysis. Requirements Engineering, 2010, 15(4): 439-458

[23] Salifu M, Yu Y, Bandara A K, Nuseibeh B. Analysing monitoring and switching problems for adaptive systems. Journal of Systems and Software, 2012, 85(12): 2829-2839

DONG Rui-Zhi, born in 1980, Ph. D. candidate. His research interests include problem frames and model checking.

PENG Xin, born in 1979, Ph. D., associate professor. His research interests include self-adaptive systems, software maintenance and software product line.

ZHAO Wen-Yun, born in 1964, professor. His research interests include software engineering and electronic commerce.

Background

This work is supported by Doctoral Fund of Ministry of Education No. 20100071110031, National High Technology Research and Development Program (863 Program) of China under Grant No. 2013AA01A605.

Nowadays, software-intensive systems are increasingly developed in an incremental and iterative manner. The requirements engineering of such systems has become a continuous cross-project and cross-product process through software lifecycle. During the process of finding solutions for incremental requirements, numerous challenges had to be addressed, including reusing the components within the current solution, reasoning about the correctness of the newly derived solutions, and locating possible risks within the preferred solutions.

Because the satisfaction of requirements within a software-intensive system results from the cooperation of software components and environmental elements, we shall take account of the relationships among the environment, the solutions and the requirements in face of the issue of incre-

mental requirements. The problem frames approach to requirements provides a basis for analyzing software problems and their context. It has been applied to deal with the problem of requirements variation and feature (requirements) evolving. However, few work pay attention to analyze the structure of the problem of incremental requirements, and let alone provide an engineering process to find solutions.

In this paper, a problem frames-based method to find solutions for the incremental requirements of a software-intensive system has been proposed. Our method takes an iterative and incremental process to seek for solutions for incremental requirements, to use SAT based techniques to reason about the correctness of the newly derived solutions, and to reuse the knowledge of typical concerns to identify possible risks of the preferred solution. A case study of a smart nursing home system has shown the effectiveness of our method. In the future, we plan to enrich our approach by adopting the change problem diagram to represent the changes imposed by the issue of incremental requirements. Moreover, we plan to conduct more empirical case studies to evaluate the effectiveness of our approach, and to develop a supporting tool for the proposed method.