目　录

插　图

第一章　绪言

1.1　研究背景

1.2　本文组织

第二章　相关工作

2.1　研究背景

2.2　本文组织

第三章　基于代码依赖关系的需求可追踪性生成方法基于代码依赖关系的需求可追踪性生成方法

3.1　引言

在软件维护和演化的过程中，一份最新的需求规约能够提供有价值的信息以辅助维护人员完成相应的软件活动。事实上，需求规约能够描述系统的主要功能与模块，解释系统实现背后的原理，帮助维护人员理解程序，并作为讨论功能更改的基础。然而，在软件演化时，维护人员通常不会及时更新需求规约。这是由于需求的更新需要付出大量的人工和时间成本。维护人员需要遍历全部的需求文档才能够定位其中需要被更新的部分，而全部的需求文档可能包含数百页的内容。因此，在面临软件维护或演化任务时，维护人员通常直接更新代码，而不更新相应的需求，很快，需求规约将会变得过时且失效。

当维护人员更新代码后，已有工作通过比较代码更新前后的差异，识别其中影响需求的变更代码元素，并利用信息检索技术计算变更代码元素与需求的相似性，从而发现近似候选过时需求。然而，系统的功能是由分布在系统间的代码协作完成的，变更部分只包含局部信息，与变更部分在结构上有依赖关系的上下文信息同样有价值。已有的工作并没有充分考虑这一点。

为了解决上述问题，我们在对过时需求的检测中引入了对代码依赖的分析。代码自身除了文本信息，还包含结构信息。我们假设在结构上具有紧密依赖关系的代码元素间，在功能上也有较高的相似性。因此通过分析代码依赖，能够得到与变更代码元素功能相似的其他代码元素，以补充与代码变更有关的信息。

本章中，我们提出一种基于代码依赖关系的过时需求自动检测方法，提高了过时需求检测时的精度。

3.2　本章小结

我们在本章定义了移动环境导致 Web 服务关联值不确定时如何计算组合服务 Skyline的问题，我们提出了安全值范围的概念，并利用安全值范围在一定程度上降低了 SAP 以及 CSKY 重新计算的次数，提高了组合服务代理的运行效率。最后，我们通过一些列实验来分析我们方法的有效性以及效率。

第四章　基于代码依赖关系的过时需求自动检测与更新推荐方法

4.1　引言

在软件维护和演化的过程中，一份最新的需求规约能够提供有价值的信息以辅助维护人员完成相应的软件活动。事实上，需求规约能够描述系统的主要功能与模块，解释系统实现背后的原理，帮助维护人员理解程序，并作为讨论功能更改的基础。然而，在软件演化时，维护人员通常不会及时更新需求规约。这是由于需求的更新需要付出大量的人工和时间成本。维护人员需要遍历全部的需求文档才能够定位其中需要被更新的部分，而全部的需求文档可能包含数百页的内容；并且维护人员需要收集和组织与变更有关的信息，以完成对需求的更新。因此，在面临软件维护或演化任务时，维护人员通常直接更新代码，而不更新相应的需求，很快，需求规约将会变得过时且失效。面对上述常见的软件活动场景，我们的工作关注于如何帮助维护人员减少检测和更新过时需求的成本。

当维护人员更新代码后，已有工作通过比较代码更新前后的差异，识别其中影响需求的变更代码元素，并利用信息检索技术计算变更代码元素与需求的相似性，从而发现近似候选过时需求。然而，系统的功能是由分布在系统间的代码协作完成的，变更部分只包含局部信息，与变更部分在结构上有依赖关系的上下文信息同样有价值。已有工作并没有充分考虑这一点。同时，对于检测得到的过时需求，维护人员可能对与变更有关的知识把握不足，需要额外进行代码分析以获得更充分的信息，已有工作没有考虑推荐与变更有关的知识以辅助维护人员完成对过时需求的更新。

为了解决上述问题，我们在对过时需求的检测中引入了对代码依赖关系的分析。代码自身除了文本信息，还包含结构信息。我们假设在结构上具有紧密依赖关系的代码元素间，在功能上也有较高的相似性。因此通过分析代码依赖，能够得到与变更代码元素功能相似的其他代码元素，以补充与代码变更有关的信息。同时，在与过时需求有关的变更代码元素中，如果一个变更代码元素的文本包含重要的概念，且此概念在代码依赖结构上反复出现，则该变更代码元素可以作为热点元素为维护人员提供与变更有关的知识。

本章中，我们提出一种基于代码依赖关系的过时需求自动检测方法，提高

了过时需求检测时的精度，并推荐与过时需求相关的变更代码元素，以辅助维护人员完成对需求的更新。

4.2　背景

在本节，我们将说明本工作的研究动机，并给出我们的问题定义。

4.2.1　研究动机

在本小节，我们将给出两个具体的例子以解释我们的研究动机。

图4.1 展示了AquaLush系统中一次真实的代码提交所涉及的代码变更，该提交为AquaLush系统新增了日志记录与展示的功能，图中*buildLogScrn()*是在本次提交中新增的函数，用于建立展示历史日志的面板，图中其余的函数在此次提交前就已经存在。在代码更新后，维护人员希望通过使用自动化工具，快速定位过时需求SRS238（需求文本内容如图 4.2 所示）。文献［］中提出的过时需求检测方法，主要关注变更代码元素*buildLogScrn()*，通过抽取该函数标识符等信息构造关键词文本，并利用信息检索技术将关键词文本与需求文本进行匹配。然而，由于信息检索技术存在词汇失配的问题，其检索的效果主要取决与检索文本的质量，只考虑变更代码元素可能导致文本的描述信息不足。例如，*buildLogScrn()*中的 “Scrn” 为单词 “screen” 的缩写，因此无法与需求文本SRSxxx中的 “screen” 相匹配，从而使得检索精度下降。

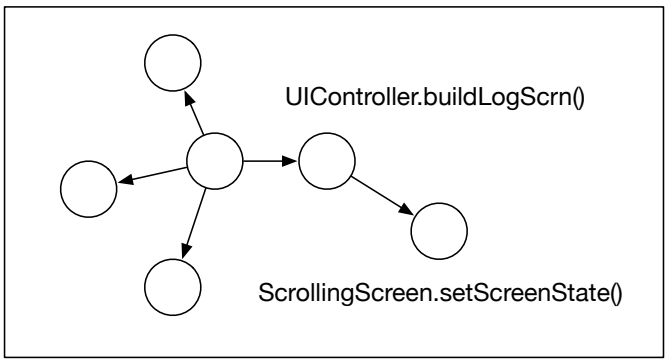


图 4.1: 与日志记录和展示功能有关的部分代码变更

我们认为，与变更代码元素在结构上有紧密依赖关系的其他代码元素同样有价值，在这个例子中，函数*setScreenState()*与且只与函数*buildLogScrn()*存在函数调用依赖，因此我们认为这两个函数之间的关系紧密，可能用于实现相似的功能，函数*setScreenState()*的文本信息能够补充与变更有关的概念。在此例中，函数*setScreenState()*包含了有效关键词 “screen” 和 “state” ，这两个概念在*buildLogScrn()*中无法体现，从而补充了与变更有关的概念，达到提高检索精度的目的。

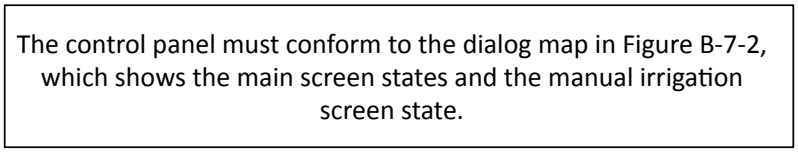


图 4.2: 需求文本SRS238的内容

4.2.2 问题定义

问题定义. 给定一个需求集合 R，该需求集合包含 *d* 个需求文本，同时给定新版本代码集合 NC 与旧版本代码集合 OC，分别包含 *m*，*n* 个代码实体（类），过时需求自动检测技术能够从 *d* 个需求文本中选择 *k* 个需求文本并排序，作为近似候选过时需求的排序，问题是：

Q1: 通过分析代码依赖关系，能否提高近似候选过时需求排序的质量？

Q2: 对于给定的过时需求OR，能否推荐与该需求相关的变更代码元素？

4.3 方法概述

4.4 代码依赖关系构成的代码变更组识别技术

4.4.1 比较代码元素的差异

在软件演化的过程中，并非所有的代码变更都会影响需求。事实上，其中许多的代码变更与缺陷修复，重构或更改系统实现细节等方面有关。Eya等人[]中提出：代码元素（包，类，函数，域）的新增或删除是影响系统外部行为，从而影响需求的代码变更。文献比较了新旧版本代码间的代码元素，以识别出新增或删除的代码元素，并过滤其中与重命名有关的变更，获得影响需求的变更代码元素。

以变更代码元素为基础，我们假定一个新增的包是有若干新增的类组成，一个新增的类是有若干新增的函数和域组成，对删除的代码元素也进行类似的处理。为描述变更代码元素，给定函数的变更集合$CM = \{CM_1, \cdots, CM_n\}$, 其中$CM_i$为新增或删除的函数，域的变更集合$CF = \{CF_1, \cdots, CF_n\}$，其中$CF_i$为新增或删除的域。

4.4.2　构造变更组

4.4.2.1　生成CDCGraph (Call Dependency with Closeness Graph)

我们利用CDCGraph以描述一个版本源代码中的函数调用依赖及其依赖的紧密度值。我们生成CDCGraph的算法包括以下两个步骤：

1. 建立函数的调用依赖：首先，我们定义CDGraph（Call Dependency Graph）为一个有序对$G =< V, E >$，其中V是代码中函数的集合，E是函数间产生的函数调用依赖集合。函数调用依赖是指，如果函数A调用了另一个函数B，那么函数A依赖于函数B。我们利用Apache BCEL库从编译后的字节码（Jar文件）中捕获函数调用依赖，对于无法直接编译的项目，我们基于JDT实现了一个工具，它能够直接从项目的Java源文件中捕获函数的调用依赖。

2. 计算函数调用依赖的紧密度值：在CDGraph中，函数之间的调用依赖只有存在或不存在两种可能，存在的所有函数调用依赖都被认为是等价的，没有对依赖的紧密程度作出区分。事实上，当函数A调用函数B的时，可能意味着以下两种情况：函数A和函数B通过协作完成某个共同的任务，两者间具有较强的依赖关系；函数A通过调用函数B实现任务间的转移，两者间具有较弱的依赖关系。为了度量函数调用依赖，我们提出紧密度值以量化函数之间的相互作用程度。我们假定，对于函数调用依赖，如果调用者的出度和被调用者的入度越小，则该函数依赖越紧密。我们定义函数调用依赖的紧密度原则的计算公式如下：

$$Closeness_e = \frac{2}{OutDegree_{e.caller} + InDegree_{e.callee}} \tag{4.1}$$

其中，$OutDegree_{e.caller}$表示调用者的出度，$InDegreee.callee$表示调用者的入度。通过赋予CDGraph中的依赖边紧密度值的方式，可得CDCGraph。

4.4.2.2　建立函数的变更域

对于变更的函数元素，我们考虑该元素的结构上下文信息，以补充与变更相关的知识。因此，我们将变更的函数元素及与其结构上紧密依赖的其他函数元素共同构成一个变更域。基于生成的CDCGraph，我们给定一个阈值k，将CDCGraph中所有紧密度值小于k的依赖边删去。剪枝后的CDCGraph被分解为若干个连通子图，每个连通子图表示一个区域，区域中的函数间被认为具有较强的相互作用程度。我们给定两个函数β和γ：第一个函数$\beta(CM_i)$，返回函数CMi在CDCGraph中所属的连通子图；第二个函数$\gamma(G_i)$，返回图G_i中所包含顶点元素的集合。对于每个变更的函数元素，它对应的变更域可表示为：

$$CR_i = \gamma(\beta(CM_i)) \tag{4.2}$$

其中，CM_i为集合CM中一个变更的函数元素。

4.4.2.3　分组策略

对于变更的函数元素，在建立其变更域后，如果孤立地考虑每一个变更域将会导致域的数量过多，并且每个变更域自身的信息可能仍不足以描述一个相对完整的系统功能。因此，我们将基于依赖关系对变更域进行合并。如果集合CM中的函数彼此之间存在函数调用依赖，则合并各个函数对应的变更域，合并后的域我们称之为一个变更组，给定集合$CG = \{CG_1, \cdots, CG_n\}$以表示变更组的集合，其中$CG_i$表示合并后的变更组。另外的，如果变更组内的函数元素使用了集合CF中变更的域元素，则将被使用的域元素加入该变更组。构造变更组的具体算法如下：

4.5　基于变更组文本与需求文本相似性的过时需求自动检测方法

4.5.1　抽取关键词以构造变更组的描述文本

在此小节，我们从变更组的代码元素中抽取关键词以构造变更组的描述文本。对于不同来源的代码元素，我们根据规则（见表N）抽取与代码元素相关的标识符，注释等信息，可以注意到，与未变更的代码元素相比，我们会为变更代码元素抽取更多相关的信息。抽词规则确立的依据是，代码元素的标识符和注释信息通常反映了需求中与系统功能相关概念。同时，与未变更的代码元素相比，我们认为变更代码元素与概念的变更有更高的相关性。对于集合CG中的每一个变更组CG_i，通过抽取关键词可以构造变更组描述文本CGD_i，$CGD_i \in CGD$。

Algorithm 1: Preprocessing of QoS Correlations

Input: service repository *SR*.

Output: service repository after preprocessing.

```
1  cwsset ← ∅;
2  forall the service ws_i ∈ SR and ws_i.CS_1 ≠ ∅ do
3      forall the service ws_j ∈ ws_i.CS_1 do
4          if ws_j.out ∩ ws_i.in ≠ ∅ then
5              ComposeServices(ws_j, ws_i);
6              remark the new composed service as cws_1;
7              add cws_1 to SR;
8              if cws_2.CS_2 ≠ ∅ then
9                  add cws_2 to cwsset;
10 while cwsset is not empty do
11     service cws ← cwsset.removeService();
12     forall the service ws_k ∈ cws.CS_2 do
13         if cws.out ∩ ws_k.in ≠ ∅ then
14             ComposeServices(cws, ws_k);
15             remark the new composed service as cws_2;
16             add cws_2 to SR;
17             if cws_2.CS_2 ≠ ∅ then
18                 add cws_2 to cwsset;
```

4.5.2　基于信息检索技术生成近似候选过时需求的排序

对于给定的变更组文本集合$CGD = \{CGD_1, \cdots, CGD_n\}$与需求集合$R = \{R_1, \cdots, R_n\}$，我们通过以下两个步骤生成近似候选过时需求的排序：（1）利用信息检索技术计算变更组文本与需求文本的相似性；（2）利用基于文本相似度值的表决算法，生成近似候选过时需求的排序。我们将在本小节余下部分详细阐述每个步骤。

1. 利用信息检索技术的计算文本相似性：基于信息检索的文本匹配技术将两个文本集合作为输入，返回文本－文本的相似度矩阵作为输出。在本方法的场景下，我们计算的是变更组文本与需求文本间的相似性，因此，我们

的输入包括变更组文本集合与需求文本集合，预期得到的输出是变更组文本－需求文本的相似度矩阵。在检索之前，我们需要对变更组文本及需求文本进行预处理。首先，对于词项来自代码的变更组文本，我们根据常用的代码命名模式（如驼峰命名法，下划线分割）进行分词操作。然后，我们对变更组文本和需求文本进行标准化处理，处理的过程包括除去特殊字符，词行还原，词根提取和去停用词。我们利用$tf-idf$[11]计算词项的权重，并利用向量空间模型计算文本相似度。

2. 基于文本相似度的表决算法：步骤一的结果是一个二维的相似度矩阵，我们基于该矩阵生成一个最终的排序列表，以向维护人员指明潜在的过时需求。我们基于文本相似度值计算最终排序列表的表决算法如下：对于每一个需求文本，累加其与变更组文本集合中各个变更组文本的相似度值，作为该需求的总得分。然后，将需求集合按需求的总得分倒叙排列，作为近似候选过时需求集合的最终排序，需求的排名越高代表该需求是过时需求的可能性越大。

4.6　实验与分析

在此小节，我们通过实验以验证本方法的有效性。接下来，将具体阐述我们的实验设置及实验结果与分析。

4.6.1　实验目标与评价指标

我们的实验目的是为了分析基于代码依赖关系的过时需求自动检测方法是否能够减少维护人员发现过时需求的成本。为了达到这一目的，我们定义了如下两个研究问题。

Q1: 分析代码依赖关系是否能够提高过时需求需求自动检测方法的精度？

这个研究问题与我们方法的第一部分有关，在x.x节中，我们验证了基于代码依赖关系的过时需求自动检测方法的有效性。

Q1的评价指标：Q1是关于从整个需求集合里检测某一特定的子集（过时需求），因此我们采用信息检索中著名的两个指标$Average\ Precision\,(AP)$和$F-measure$以评价近似候选过时需求排序的质量。

$$AP=\frac{\sum_{r=1}^{N}\left(Precision\,(r)\times isRelevant\,(r)\right)}{|RelevantDocuments|} \tag{4.3}$$

其中，r表示目标文本在有序文本列表中的排序，$Precision\,(r)$表示前r个排序的准确率，$isRelevant\,()$是一个二值函数，如果文本相关返回1，无关则返回0，

N表示文本的总数。同时，$F-measure$是准确率和召回率的调和平均值，计算方式如下：

$$F = \frac{2}{\frac{1}{R} + \frac{1}{P}} \tag{4.4}$$

其中，R表示召回率，P表示准确率。我们通过统计显著性检验以验证我们方法能够在不同的截止排序n上提高其$F-measure$值。由于过时需求的总数对于每个检测方法是一致的，因此我们采用Wilcoxon秩和检验[]以检验如下的两个零假设（null hypothesis）：H_0：在代码提交版本间，分析代码依赖关系没有显著的提高过时需求需求自动检测方法的精度。

H_1：在代码发布版本间，分析代码依赖关系没有显著的提高过时需求需求自动检测方法的精度。

我们采用$p-value$显著性差异水平0.05作为衡量检验结果的标准。除了$p-value$以外，我们关注的另一方面是在实践中不同方法精度间的差异幅度。为此，我们采用$Cliff.s\ Delta\ d$[68]，一种应用于序数的无参效果量（$effective\ size$）来衡量实验组与对照组的差异（$p-value$表明显著性是否存在，$effective\ size$表明实践的显著程度）。$Cliff.s\ Delta$的范围在-1和1之间，$d < 0.33$表示显著性较小，$0.33 \leq d < 0.474$表示显著性一般，$d \geq 0.474$表示显著性较大。

4.6.2　研究案例与实验设置

1. iTrust

   我们第一个研究案例是医疗服务项目iTrust[16]，iTrust是一个医疗数据管理系统，它是由北卡罗莱纳州立大学开发并维护的一个用于教学目的的系统。iTrust包含多个发布的代码版本和一个基于wiki的规约说明，规约说明中包括功能性需求，非功能性需求，术语表等信息。iTrust是由Java和Java Server Page组合开发的Web应用。在我们的实验中，需求部分我们选用iTrust中的39个功能性需求，每个功能性需求是详细且定义良好的用例(use case)。在代码部分，由于我们的原型工具暂时只支持对Java代码的分析，因此我们只考虑iTrust中Java部分的源码。我们使用iTrust版本10（发布日期：2010.8.18）和版本11（发布日期：2011.1.7）作为两个发布的代码版本。Eya在文献[]中通过人工比较这两个版本源代码的方式，识别了14个概念变更，我们选取了其中10个影响需求的变更作为代码提交。表N中展示了我们所选取的代码变更，变更的描述，以及变更所影响的需求。

表 4.1: iTrust系统中的代码变更及变更所影响的需求

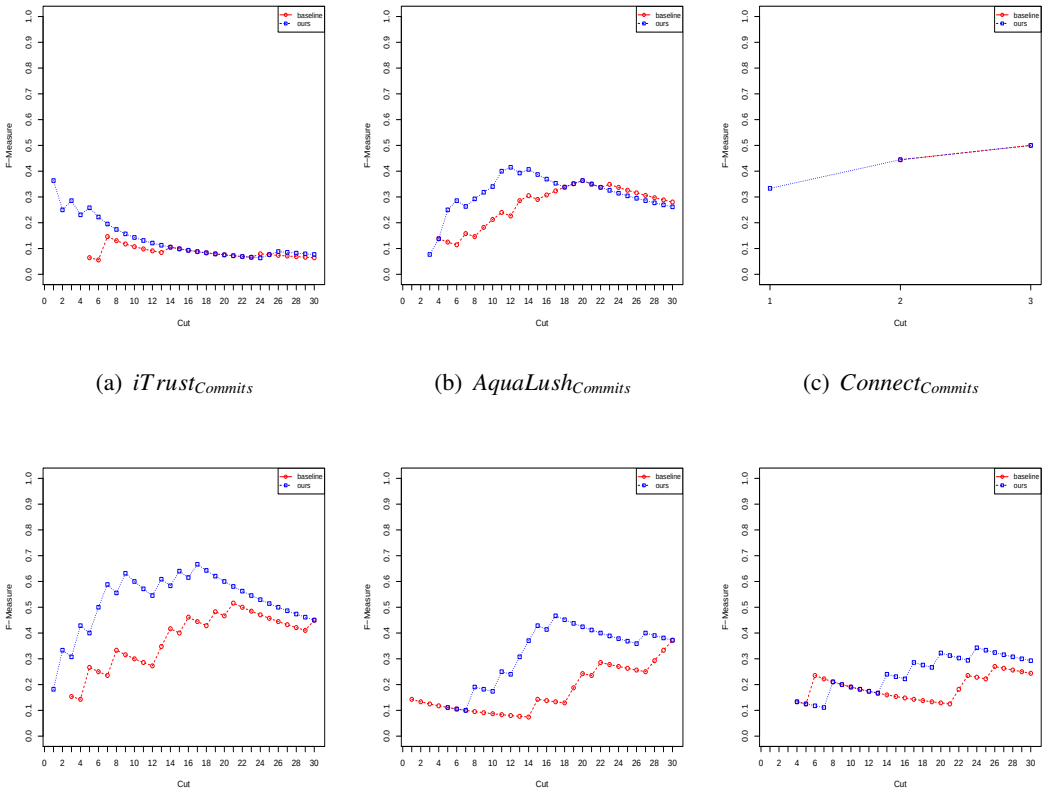
Change	Change description	Impacted use cases
Change 1	Reason code	UC15, UC37
Change 2	Uploading photo	UC4
Change 3	Remote monitoring (added height, weight, etc.)	UC34
Change 4	Remote monitoring (get patient data by type)	UC34
Change 5	Display patient’s monitoring HCP	UC34
Change 6	Office visit form (added orc and comment)	UC11
Change 7	Enabling appointment editing	UC22
Change 8	Login (added captcha and attempts)	UC3
Change 9	Weight/height charting	UC10
Change 10	Logging (added logs)	UC5

2. AquaLush

我们第二个研究案例是由软件控制的灌溉管理系统[]，可以基于用户提供的湿度、用水量等参数自动调整灌溉输出，AquaLush最初作为一个解释性示例在《软件设计导论》[14]一书中被提出。AquaLush由Java语言实现，代码量在1.1万行左右，同时，AquaLush包含自然语言描述的结构化需求规约。网页[]包含AquaLush的代码，需求和基线等相关。在需求的部分，我们的实验提取了需求规约中文本格式的337个需求说明作为需求规约集合；在代码的部分，Eya在文献[]中开发了AquaLush的另一个版本，相比原版本，新版本包含了8次代码提交（3次与系统功能有关和5次与缺陷修复有关）。我们选取了3次与系统功能有关的代码提交，并将上述两个版本作为两个代码的发布版本。

3. Connect

我们第三个研究案例是用于医疗机构间交换医疗数据的开源系统Connect[]。Connect项目包含大量相关的文档，如设计文档，使用手册，问题追踪，变更请求，发布记录，各代码版本的需求，需求跟踪矩阵等。Connect由Java语言实现，并包含历史提交信息。在本实验中，我们采用了主项目部分的Java代码，它位于代码仓库的product/production目录下，代码量在28万行左右。在需求的部分，我们采用了需求跟踪矩阵及发布记录中Jira单号，其中需求跟踪矩阵关联了需求与Jira问题单号的关系，通过它们之间的关



(a) $iTrust_{Commits}$　(b) $AquaLush_{Commits}$　(c) $Connect_{Commits}$

(d) $iTrust_{Releases}$　(e) $AquaLush_{Releases}$　(f) $Connect_{Releases}$

图 4.3: F-measure/cut curves for iTrust, AquaLush and Connect.

联，能够便于我们发现与相应需求有关的代码提交。

4.6.3　实验结果与结果分析

4.7　本章小结

我们在本章定义了移动环境导致 Web 服务关联值不确定时如何计算组合服务 Skyline的问题，我们提出了安全值范围的概念，并利用安全值范围在一定程度上降低了 SAP 以及 CSKY 重新计算的次数，提高了组合服务代理的运行效率。最后，我们通过一些列实验来分析我们方法的有效性以及效率。

第五章　总结与展望

5.1　工作总结

本文针对服务之间存在 QoS 关联情况下，组合服务 Skyline 如何计算的问题，提出了一套高效的解决方案，并考虑了移动环境下，服务之间的 QoS 关联值会动态变化的情况，针对该问题同样提出一套高效的解决方案。具体来说，本文首先给出了一套支持 QoS 关联的 Web 服务模型，基于该模型提出了一种支持 QoS 关联的组合服务 Skyline 计算方法，并设计出若干剪枝规则，加速该方法的执行效率。然后针对移动 Web 服务的使用场景，提出了安全值范围的概念，基于安全值范围，我们降低了组合服务 Skyline 在 QoS 关联值动态变动情况下的计算代价，除此之外我们还给出了安全值范围的计算和更新方法。最后，通过一系列实验，验证了我们方法的有效性和正确性。

5.2　研究展望

针对存在 QoS 关联的场景，本文虽然可以高效的解决 QoS 关联存在于一个质量属性上的情况，但是对于 QoS 关联存在于多个质量属性的情况并不能很好的解决。在移动环境下，虽然可以在一定程度上提高组合服务 Skyline 的计算效率，但是从实验结果来看，仍有很大的提升空间。文中仅考虑了 QoS 关联值在移动环境下动态变化的情况，假设了QoS值在移动环境下不发生变化，但实际情况两者都会发生变化。此外，本文只考虑抽象组合服务为顺序结构。我们希望以后的工作可以针对以上不足进行扩充，并解决以上问题。

简历与科研成果

基本情况 杜宇，男，汉族，1991 年 02 月出生，江苏省徐州市睢宁县人。

教育背景

2012.9～2015.6 南京大学计算机科学与技术系 硕士

2008.9～2012.6 南京大学金陵学院软件工程 本科

攻读硕士学位期间发表的论文

1. **Yu Du**, Hao Hu, Wei Song, Junhua Ding and Jian Lü. Efficient Computing Composite Service Skyline with QoS Correlations. In *Proceedings of International Conference on Services Computing*, 2015, Accepted.
2. **Yu Du**, Hao Hu, Wei Song, Yuhao Gong and Jian Lü. Safety-Range Aware Mobile Composite Service Skyline. In *Proceedings of International Conference on Mobile Services*, 2015, Accepted.

攻读硕士学位期间申请的专利

1. 胡昊，宋巍，葛季栋，吕建，**杜宇**，“一种计算具有QoS关联关系的QoS最优组合服务限定范围的方法”，申请号：201510168726.0，已授权。

攻读硕士学位期间参与的科研课题

1. 国家973计划：持续演进的自适应网构软件模型、方法及服务质量保障。项目编号No.2015CB352202。

参考文献