软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2011,22(12):2866−2878 [doi: 10.3724/SP.J.1001.2011.03992]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

一种网络化移动应用部署方案优化方法*

张晓薇[1,2], 曹东刚[1,2+], 陈向群[1,2], 梅 宏[1,2]

[1](北京大学 计算机科学与技术系,北京 100871)

[2](高可信软件技术教育部重点实验室(北京大学),北京 100871)

Deployment Solution Optimization for Mobile Network Applications

ZHANG Xiao-Wei[1,2], CAO Dong-Gang[1,2+], CHEN Xiang-Qun[1,2], MEI Hong[1,2]

[1](Department of Computer Science and Technology, Peking University, Beijing 100871, China)

[2](Key Laboratory of High Confidence Software Technologies of Ministry of Education (Peking University), Beijing 100871, China)

+ Corresponding author: E-mail: caodg@sei.pku.edu.cn

Zhang XW, Cao DG, Chen XQ, Mei H. Deployment solution optimization for mobile network applications. *Journal of Software*, 2011,22(12):2866−2878. http://www.jos.org.cn/1000-9825/3992.htm

Abstract: This paper proposes an integrated approach to facilitate mobile application development and deployment from software architecture perspective. It models, in multiple dimensions, the device parameters (like CPU, memory, screen, communication module), user preferences (like energy or performance preference), and QoS requirements (like frequency of interaction, average size of event) at architectural level. This approach will generate personalized deployment plans to meet specific requirements of mobile users. The case study and experiment results show that this approach effectively facilitates development and deployment, and improves the customizability of mobile network applications.

Key words: software architecture; software customization; user profile; QoS; mobile network application

摘 要: 从软件体系结构角度出发,从满足移动用户个性化需求入手,提出一种便于移动应用开发和部署的整合方案.该方案从设备属性(如 CPU、内存、屏幕、通信模式等)、用户偏好(如对能耗和性能的偏好等)以及 QoS 需求(如交互频率、安全、实时性等)等多重维度入手,采用体系结构驱动的方法对应用进行建模,并生成满足用户个性化需求的部署方案,从而有效提高了移动应用的可配置性.实验结果表明,该方案可以有效地方便移动应用的开发及部署,提高应用与用户需求的契合程度,改善网络化移动应用的可配置性.

关键词: 软件体系结构;软件可配置性;用户配置信息;服务质量;网络化移动应用

中图法分类号: TP311 **文献标识码**: A

手持设备和无线网络的普及,为移动用户提供了丰富的软件和服务选择.通过无线网络,用户可以使用服务器提供的多种服务和计算能力.然而,不同的移动用户具有不同的软/硬件平台特性、不同的用户偏好和 QoS 要求,因此,为其提供的移动软件和服务应尽可能地满足用户的个性化需求,即提供具有高可配置性的移动软件和

* 基金项目: 国家自然科学基金(61073020, 60821003); 国家重点基础研究发展计划(973)(2011CB302604)
收稿时间: 2010-09-15; 修改时间: 2010-12-09; 定稿时间: 2011-01-31
CNKI 网络优先出版 2011-05-26 14:01, http://www.cnki.net/kcms/detail/11.2560.tp.20110526.1401.007.html

服务.

以“移动相册”[1]为例,有蓝牙通信功能的高端移动客户端应具备蓝牙传输照片这一功能模块,而低端设备则不必具有此项功能.另外,用户偏好对于照片的显示亦具有重要影响.例如,对希望节约能耗的用户,其客户端应具有图片压缩模块将彩色图片转换为灰度图片以减少对能源的消耗.再者,用户可能会要求构件间传输的事件大小不超过 100 字节以保证整体性能,因此,客户端应具备满足这一用户 QoS 需求的相应构件.由该实例可以看出,网络化移动应用的主要特点包括:

- 应用设备专属性:移动设备具有专属性特征,因此,网络化移动应用以满足移动终端用户需求为目的,为其提供软件服务;
- 应用软/硬件平台多样性:不同的移动设备具有不同的软/硬件平台特性,因此,网络化移动应用既应适应不同平台的软/硬件限制,又应充分发挥其平台的软/硬件能力;
- 用户偏好个人化:不同用户对于网络化移动应用具有不同的使用偏好,用户偏好的个人化亦需要网络化移动应用给予针对性满足,在具体部署方案中增加对用户偏好的相应处理;
- 应用 QoS 需求多样化:设备软/硬件特性的不同以及用户偏好的个性化,使得不同用户对网络化移动应用的 QoS 需求亦有所不同.因此,网络化移动应用部署方案应能满足不同的应用 QoS 需求;
- 应用环境动态性:用户地理位置的移动性以及网络/电源等硬件资源的动态性,使得网络化移动应用的执行环境较为多变,因此,网络化移动应用应满足并适应执行环境的动态性,针对不同的应用环境为用户提供满足其需求的软件服务.

因此,为了更好地服务用户,具体的网络化移动应用部署方案应尽可能地适应并满足平台多样性、用户偏好个人化、QoS 需求多样化以及应用环境动态性等应用特点.而如何自动生成满足上述应用特点、良构的网络化移动应用部署方案,成为亟待解决的问题.

软件体系结构[2,3]主要关注于以构件、连接子和以配置为中心的系统结构设计与规约,可以有效解决复杂、动态系统的开发,因为它可以有效辅助系统建模,对非功能属性(如性能、移动性、可靠性、安全性)进行建模和分析[4−6].然而在网络化移动应用领域,软件体系结构、设备属性、用户偏好以及 QoS 需求之间的关系尚未系统阐述清楚,如何利用软件体系结构来有效提高应用的可配置性尚需进一步探讨.因此,本文试图将以上 4 个方面整合起来形成一种对网络化移动应用进行设计、分析、部署、监控和重配置的有效方法 ADCA(architecture-driven customization approach).ADCA 方法可以产生满足用户不同需求的应用部署方案,从而有效改善应用的可配置性.该方法的主要组成部分如下所示:

- 基于体系结构的应用建模.在描述构件、连接子和配置时,考虑个性化用户需求.利用我们提出的构件描述语言 NanoADL[7,8]对软件体系结构元素和个性化用户需求加以描述,这一良构描述有利于具有高可配置性的移动软件的自动生成;
- 非功能属性分析和部署方案的生成.以前面提到的应用建模结果作为输入,分析不同部署方案对于用户需求的满足情况,进而以用户满意度为依据选择最优的部署方案;
- 运行时监控和重配置.在系统运行时,检查与用户满意度相关的主要属性指标.一方面,系统运行环境的动态变化会导致应用部署方案与用户需求的不一致,从而带来应用重配置的需求,例如在“移动相册”实例中,当网络变得非常不稳定时,移动用户会希望通过数据缓存模块来保证应用的连续可用性;另一方面,同一用户的偏好亦会随时间发生变化,例如随着电源的不断消耗,用户对于节约能耗的偏好会增强,从而带来对应用部署方案重配置的需求.

本文第 1 节简要介绍与本文相关的研究工作.第 2 节介绍体系结构驱动的提高应用可配置性的方法,并将该方法应用于具体的网络化移动应用实例.第 3 节对算法实验效果和重配置实验结果进行分析.第 4 节对全文工作进行总结,并提出进一步工作的研究设想.

1 相关工作

与本文相关的研究工作可以分为以下 4 类:针对移动环境的软件体系结构驱动的中间件技术研究、用户信息描述技术研究、用户偏好信息获取技术研究以及移动应用产品线技术研究.

Prism-MW[9,10]主要关注于分布式、资源受限的、异步移动环境的中间件技术研究.它利用“类”的概念对软件体系结构要素加以表达,同时利用“类”中的“方法”对软件体系结构要素完成创建、销毁等管理操作,这样建立起体系结构与其实现之间的直接映射.为了充分利用其运行时动态增删构件的能力,本文在 Prism-MW 的中间件平台基础上建立对软件体系结构要素的非功能属性描述和建模,尤其是与用户个性化需求相关的属性建模,从而在良构模型基础上生成满足用户需求的部署方案.

CC/PP[11]是一种描述用户设备能力的通用、可扩展框架,其以 RDF(resource description format)的形式对用户信息加以描述.其设计目的在于,根据用户的设备能力,完成内容、服务与用户设备的适应与匹配.本文对 CC/PP 进行扩展,使其可以对个性化用户需求进行表述,使得移动服务器可以为移动用户生成合适的软件部署方案或在线服务.

用户偏好信息的获取并不容易,原因是:一方面,对此类主观信息的量化存在困难;另一方面,人们对偏好信息的描述大多采用自然语言方式,而非机器可理解的方式.近年来,一些研究人员关注于利用静态分析或机器学习技术来量化用户偏好.文献[12]针对可穿戴的嵌入式系统,通过对个人状态以及用户与系统交互方式的识别,利用机器学习的技术完成对用户偏好信息的获取.本文并不关注于用户偏好信息的具体获取机制研究,而是关注如何利用用户偏好信息提高网络化移动应用的可配置性,为用户提供更好的服务.

软件产品线技术(software product line)对于处理由移动设备多样性带来的软件开发问题是一种常用的解决方案,其从模块建模、模块实现和模块测试这 3 个方面为不同的移动设备生成合适的软件方案[13].软件产品线技术主要关注于对复杂的软件家族产品的管理,关注于如何提高不同产品模块的复用性等问题.本文则更多地从用户个性化需求角度入手,综合考虑设备特性和用户偏好以及 QoS 要求,以生成符合用户要求的软件部署方案或在线服务为目标.另外,网络化移动应用运行环境的多变性以及移动用户需求偏好的可变性,对于运行时监控以及重配置提出了更高的要求.相比于软件产品线技术,基于软件体系结构的方法可以更灵活地处理动态性带来的相关问题.

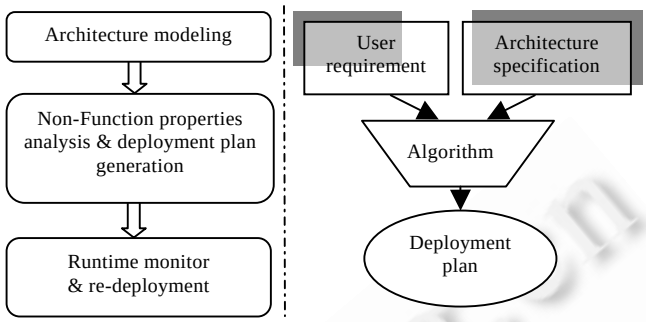
2 软件体系结构驱动的网络化移动应用部署方案的优化方法

本文提出的软件体系结构驱动的优化移动应用部署方案的方法 ADCA 覆盖多个软件开发生命周期:从体系结构建模到运行时重配置.图 1 展示了 ADCA 方法的概览,其中,图 1(a)展示了应用 ADCA 方法的主要过程,图 1(b)展示了其中部署方案生成这一阶段所用方法的主要步骤.

体系结构建模阶段.利用 NanoADL[7,8]从功能需求到非功能限制进行整体描述,通过 NanoADL 建立起应用功能模块与其非功能属性之间的关系,为后续配置应用部署方案提供良好的工作基础.NanoADL 针对资源受限环境的移动应用,对移动应用的构件、连接子和配置进行描述,同时对各类非功能属性亦进行规约.

非功能属性分析及部署方案生成阶段.如图 1(b)所示,基于个性化用户需求以及体系结构建模阶段生成的体系结构规约生成满足用户需要的部署方案.部署方案生成算法对用户个性化需求进行分析,并在合理的时间范围内生成最优的或优化的部署方案.

运行时监控及重配置阶段.监控运行时客户端执行环境和个性化用户需求的重要变化,并根据实际环境确定是否进行部署方案的重新生成,进而进行客户端应用的重配置.



(a) Overall process (b) Process of deployment plan generation

(a) ADCA 主要过程概览 (b) 部署方案生成主要步骤

Fig.1 Overview of ADCA

图 1 ADCA 概览

在本文提出的 ADCA 方法中,一个关键问题在于如何将需求层面的用户个性化需求映射到各个构件上,在应用的设计层面反映需求要素.为此,本文从用户个性化需求描述、构件/连接子规约以及用户满意程度定性度量这 3 个方面对此加以解决:

(1) 用户个性化需求描述.从“设备属性要求”、“用户偏好”以及“QoS 需求”等 3 方面描述用户需求,并对其进行半形式化抽象,为用户满意程度度量打下基础;

(2) 构件/连接子规约.对构件/连接子的功能及非功能属性进行全面规约,以非功能属性为桥梁建立用户设计层面与需求层面之间的关联,并且通过对构件/连接子规约的半形式化抽象形成用户满意程度度量的基石;

(3) 用户满意程度度量.通过比较用户需求和构件/连接子非功能属性规约,定性度量部署方案与用户需求之间的匹配程度.

2.1 个性化用户信息描述

本文以“个性化用户信息”来描述影响移动应用可配置性的主要因素,包括:设备属性,如硬件信息(CPU、内存、屏幕、通讯模块等)、软件平台属性(操作系统、Java 虚拟机、HTML 版本等);用户偏好,如对节约能耗的偏好、对执行性能的偏好等[1];QoS 需求,如对模块交互频率、安全、实时性等方面的需求.

本文扩展 CC/PP 对“个性化用户信息”进行描述,并通过对相似属性进行分组实行层次化管理.例如:将“硬件能力”作为一个信息分组,其中包含内存、CPU、通信模块等属性信息;“软件平台”则可作为另一个信息分组,包含 OS 版本信息、Java 虚拟机版本信息等.对于每一个属性信息通过二元组〈属性名,属性值〉进行表示.本文以 RDF 形式来描述和存储“个性化用户信息”,如下所示:

```
〈HardwarePlatform〉
  〈Defaults
  〈Vendor=“Nokia”
   Model=“2160”
   Type=“PDA”
   ScreenSize=“800×600”
   CPU=“PPC”
   Keyboard=“Yes”
   Memory=“16MB”
   Bluetooth=“Yes”
   Speaker=“Yes”/〉
  〈Modifications
   Memory=“32mB”/〉
〈/HardwarePlatform〉
```

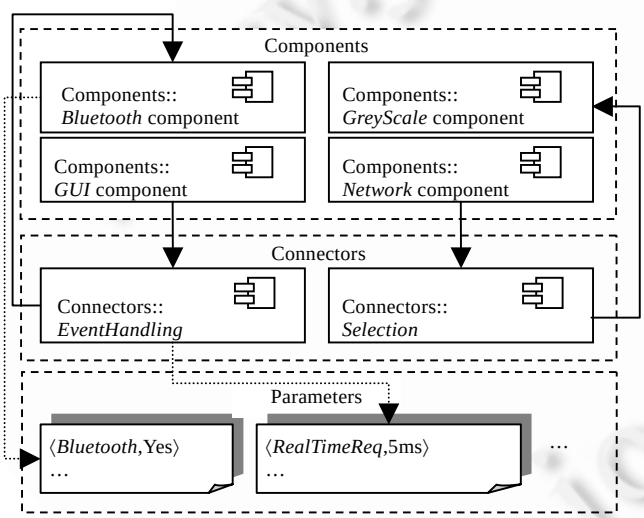
```
〈UserPreference〉
  〈Defaults
    PreferenceToEnergy =“0.5”
    PreferenceToPerformance=“0.5”/〉
〈Modifications
    PreferenceToEnergy =“0.8”
    PreferenceToPerformance=“0.2”/〉
〈/UserPreference〉
```

在前面示例中,每一组个性化信息均由默认值来描述一般环境的属性设置,同时可通过 *Modification* 来表达定制化的用户信息.例如,*UserPreference* 组别的 *PreferenceToEnergy* 属性默认为 0.5,根据用户实际需要,可将其修改为 0.8.另外,采用 RDF 形式可以方便地根据应用需求对“个性化用户信息”进行扩展.例如,对于移动多机游戏应用而言,可能需要对硬件通信模块的蓝牙设备加以描述,只需在 *HardwarePlatform* 组别增加 *Bluetooth* 标签即可.

2.2 体系结构建模

为了便于个性化部署方案的生成,本文从应用整体结构,到应用功能模块,到“个性化用户信息”进行全面建模,将构件描述与“个性化用户信息”描述进行整合,这些个性化参数作为部署方案组成部分选择的主要依据,同时亦可作为运行时是否进行重配置的检测依据.例如,若实时性是衡量应用质量的重要方面,则事件或请求的响应时间以及构件间的时序关系应被建模为构件或连接子的重要属性参数.

“移动相册”是一个典型的网络化移动应用[1],图 2(a)说明了该应用的主要结构,它由 4 个主要构件组成,分别为功能模块 *Bluetooth*,*GreyScale*,*GUI* 和 *Network*,通过 *EventHandling* 和 *Selection* 这两个连接子将这些构件连接起来.另外,构件和连接子亦可具有各自的属性特征描述,例如,*RealTimeReq* 用来描述连接子 *EventHandling* 对于实时性的要求.图 2(b)则利用 NanoADL 这一体系结构描述语言对应用加以描述.



(a) Overview of mobile album

(a) 移动相册概览

```
⟨Component⟩
    ⟨name⟩Bluetooth⟨/name⟩
    ⟨provideinterface⟩Send⟨/provideinterface⟩
    ⟨provideinterface⟩Receive⟨/provideinterface⟩
    ⟨requireinterface⟩Decode⟨/requireinterface⟩
    ⟨DeviceParameter⟩
    ⟨Bluetooth⟩“Yes”⟨/Bluetooth⟩
    ⟨/DeviceParameter⟩
⟨/Component⟩
⟨Component⟩
    ⟨name⟩GreyScale⟨/name⟩
    ⟨provideinterface⟩
        ConvertFormat
    ⟨/provideinterface⟩
⟨/Component⟩
….
⟨Connector⟩
    ⟨name⟩EventHandling⟨/name⟩
    ⟨connectedcomponent⟩
        GUI
    ⟨/connectedcomponent⟩
    ⟨connectedcomponent⟩
        Bluetooth
    ⟨/connectedcomponent⟩
    ⟨QoSParameter⟩
    ⟨AvgEventSize⟩“500b”⟨/AvgEventSize⟩
    ⟨/QoSParameter⟩
⟨/Connector⟩
…
```

(b) Architecture specification based on NanoADL

(b) 基于 NanoADL 的体系结构规约

Fig.2 Architecture of mobile album

图 2 移动相册的体系结构

基于这样的体系结构建模方法,本文对网络化移动应用进行半形式化抽象和表达,下面展示了网络化移动应用的半形式化模型:

构件集合 *CM*,构件参数集合 *CMP*,函数 $cmPara:CM\times CMP\rightarrow R$

连接子集合 *CN*,连接子参数集合 *CNP*,函数 $cnPara:CN\times CMP\rightarrow R$

部署方案集合 $DeplSpace=\{d_1,d_2,\ldots,d_n\}, n=|CM|!$

个性化用户信息集合 *UPA*,函数 *UPAPara*:*UPA*→*R*

个性化用户信息优先级集合 *PrioUPA*,*PrioUPA*⊆Z^+,函数 *UPAPrio*:*UPA*→*PrioUPA*

属性限制条件集合 *AC*,函数 *acGuaranteed*:*AC*×*DeplSpace*→*R*

函数 *userSatisfaction*:*UPA*×*DeplSpace*→*R*

其中,

- *CM* 表示构件集合,其中的构件由若干非功能属性参数加以描述,如内存要求、电源要求、JVM 版本要求等.这些非功能属性构成构件参数集合 *CMP*.函数 *cmPara*(*cm*,*cmp*)表示构件非功能属性与其属性值之间的映射关系;
- *CN* 表示连接子集合,其中的连接子由若干参数加以限制和描述,如构件交互频率、交互实时性等.这些参数构成连接子参数集合 *CNP*.函数 *cnPara*(*cn*,*cnp*)将连接子参数映射为具体的参数值;
- *UPA* 表示个性化用户信息集合,其包含了描述用户个性化需求的若干参数,如各种硬件特性、软件平台、用户偏好和 QoS 需求等.函数 *UPAPara*(*upa*)表示个性化用户需求属性与属性值之间的映射关系.另外,在不同的应用环境中,这些个性化需求的优先级亦有所不同.例如,当移动设备电量过低时,节能需求的优先级应高于其他需求.这里,*PrioUPA* 是个性化需求优先级的集合,是正整数集合的子集.函数 *UPAPrio*(*upa*)则为每个个性化用户需求分配一个优先级;
- *DeplSpace* 表示部署方案集合,其中,每个部署方案由若干构件和相应的连接子构成.可能的部署方案数目为|*CM*|!,其中,|*CM*|表示构件集合 *CM* 中的构件数目.函数 *userSatisfaction*(*upa*,*depl*)表示用户在某一需求上对部署方案的满意程度;
- 属性限制条件集合 *AC* 包含了对部署方案组成部分的限制条件.函数 *acGuaranteed*(*ac*,*depl*)反映部署方案与限制条件是否匹配.例如,限制条件"HTML 版本必须为 1.0 以上"将对部署方案中的构成部分加以约束,若某构件实现适用的 HTML 版本低于 1.0,则该构件与这一限制条件矛盾,不应入选最终部署方案.

另外,半形式化模型中的若干元素具有可扩展性,如 *UPA*,*AC* 等,可以根据应用需要对其进行定制和扩展.

本文基于上述的网络化移动应用半形式化模型来解决生成满足用户个性化需求的部署方案问题.为此,本文以用户满意度为标准选择具体的部署方案.下面展示了生成部署方案这一问题的形式化表述.

确定满足用户个性化需求的最优部署方案 *d*:

最大化

$$satisfaction(d)=\sum_{i=1}^{|UPA|}userSatisfaction(upa_i,d)\times UPAPrio(upa_i) \tag{1}$$

其中,

$$\left.\begin{aligned}userSatisfaction(upa_i,d)=&\sum_{element_j\in d\wedge element_j\in CM}\left(\sum_{para\in CMP}upaCMPMatch(upa_i,para,element_j)\right)+\\&\sum_{element_j\in d\wedge element_j\in CN}\left(\sum_{para\in CNP}upaCNPMatch(upa_i,para,element_j)\right)\end{aligned}\right\} \tag{2}$$

$$upaCMPMatch(upa_i,para,element_j)=\begin{cases}1, & \text{if } cmPara(element_j,para)-UPAPara(upa_i)>0\\0, & \text{if } cmPara(element_j,para)-UPAPara(upa_i)<0\end{cases} \tag{3}$$

$$upaCNPMatch(upa_i,para,element_j)=\begin{cases}1, & \text{if } cnPara(element_j,para)-UPAPara(upa_i)>0\\0, & \text{if } cnPara(element_j,para)-UPAPara(upa_i)<0\end{cases} \tag{4}$$

另外,部署方案 *d* 需满足以下限制条件:

$$\forall constraint\in AC, acGuaranteed(constraint,d)=1 \tag{5}$$

$$acGuaranteed(constraint,d)=\begin{cases}1, & \text{if } cmPara(cm,constaint)-UPAPara(constraint)>0,\forall cm\in d\\ 0, & \text{otherwise}\end{cases}.$$

函数 *satisfaction*(*depl*)量化用户对整个部署方案的满足程度.这里,我们从部署方案对每个用户个性化需求的满足程度(即 *userSatisfaction*)以及该需求的优先级(即 *UPAPrio*)两个方面来进行量化,见式(1).在量化 *userSatisfaction* 时,主要从构成部署方案的各构件和连接子与各个性化需求的匹配程度加以衡量(即 *upaCMPMatch* 和 *upaCNPMatch*),见式(2).构件(或连接子)的非功能属性与用户个性化需求的匹配由函数 *upaCMPMatch*(或 *upaCNPMatch*)加以表达,其函数值根据是否满足需求被映射为 0 或 1,见式(3)或式(4).在最大化 *satisfaction*(*depl*)的过程中,亦需要满足各类限制条件,见式(5).综上所述,根据用户需求生成部署方案这一问题可以抽象为具有限制条件的整数规划问题.

2.3 部署方案生成算法

在生成部署方案之前,还需要对模型中的非功能需求进行语义映射,因为不同的应用环境中非功能属性的语义信息有所不同.例如,在大多数移动 Web 应用中,带宽 40kb/s 可以被量化为“高质网络条件”;然而在移动多媒体应用中,这一带宽只能被量化为“普通网络条件”.因此,有必要建立起非功能属性与其语义之间的映射关系.

在第 2.2 节部署方案生成形式化模型中,为了最大化用户满意度,需要在多个因素之间进行权衡和比较;另外,应用于部署方案生成的算法在复杂度和效率方面亦需要满足应用场景的需求.目前,已有若干算法可以用来处理多变量优化问题.本文综合贪心算法和混合整数非线性规划算法(mixed-integer nonlinear programming,简称 MINLP)来满足不同应用场景的部署方案生成需求.

贪心算法通过作一系列的选择给出某一问题的最优解,它所作出的每一个选择是当前状态下的最好选择:

- 能否生成最优方案的讨论:由于本文的部署方案优化问题与 0-1 背包问题类似,具有贪心选择性和最优子结构性质,因此在时间充足的条件下,可以生成最优部署方案.当时间限制条件较紧时,亦可以生成当前最优方案(可能不是全局最优);
- 算法适用性讨论:贪心算法较为适合计算部署方案的可用时间较为有限这一场景.例如,第 2.2 节中提到的“移动相册”应用场景,移动用户从远程服务器下载满足用户需求的软件版本.若部署方案生成算法耗时过多,用户可能无法忍受过长的等待时间,因此,宜采用贪心算法对部署方案进行迭代优化,当设定好的等待时限到达时,将当前的最优部署方案(而非全局最优)提供给用户;
- 算法主要过程:
  1. 对可选构件(包括由其附带的可选连接子)依据每一个用户个性化需求属性 upa_i 计算:

  [*upaCMPMatch*(upa_i,*para*,可选构件)+*upaCNPMatch*(upa_i,*para*,可选构件的附带连接子)]×*UPAPrio*()

  并将计算结果根据 upa_i 进行加总,并对加总数值进行降序排序;
  2. 按照步骤 1 中的排序结果从高到低依次选择可选构件加入.

本文提出的部署方案优化问题属于混合整数非线性规划(MINLP)范畴,因此采用 Illinois 大学开发的 Baron(branch-and-reduce optimization navigator)[14,15]来解决部署方案的优化问题:

- 能否生成最优方案的讨论:由于部署方案优化函数 *satisfaction*(*d*)是凸函数,因此可以利用 Baron 为移动用户生成全局最优的部署方案.在不限制用时的条件下,MINLP 算法可以生成全局最优的部署方案,但是算法的中间计算结果并不一定是局部最优解,因此使用 MINLP 算法解决优化问题应该为其提供较为宽松的时间约束;
- 算法适用性讨论:MINLP 算法适用于生成部署方案无时间限制这一场景.仍然以“移动相册”为例,当用户已经从服务器获得所需软件后,随着使用时间的推移,电源供给随之减少,因此用户对节约能耗的偏好会增强,之前的软件版本可能需要有所改变以满足用户的当前需求.在此应用场景下,重新生成部署方案的请求并不像适用于贪心算法的场景那样严格要求算法执行时间,因此可以采用相对耗时的 MINLP 这种解决多变量优化问题的通用技术,来为用户生成最优部署方案.

另外,为了将优化问题表示为整数规划问题,从而利用贪心算法或 MINLP 算法达到优化应用部署方案的目

的,需要定义决策变量 sc_i,该变量表示构件 i 是否被包含在部署方案 $depl$ 中,即

$$sc_i=\begin{cases}1, & \text{若构件}i\text{包含在部署方案}depl\text{中}\\ 0, & \text{若构件}i\text{不包含在部署方案}depl\text{中}\end{cases}.$$

部署方案生成算法将完成对 sc_i 的赋值,使得 $satisfaction(depl)$最大.然而 MINLP 对于非凸性函数不存在最优解的情况无法提供有效处理[16],当 $satisfaction(depl)$函数为非凸性时,本文采用贪心算法为用户生成满足其个性化需要的优化部署方案.

2.4 运行时监控及重配置

为了适应多变的应用执行环境以及满足不断变化的用户需求,需要在运行时监控移动应用所处环境信息,并且在环境变化时为用户获得适当的新部署方案.图 3 展示了部署方案重配置的主要过程.重配置时机的选择是部署方案重配置中的一个关键问题:若触发重配置时机过于频繁,将会导致系统资源的浪费;而重配置时机触发频率过低,将使得应用系统重配置机会过少,造成配置方案对移动应用环境的调整和适应不足.当运行时监控(monitor)检测到用户个性化需求或用户所处环境发生显著变化时,移动客户端(mobile client)连同当前“个性化用户信息”向服务器端(server)发送重配置请求(re-deployment request).服务器在接收到请求后,将重新计算并生成部署方案,并进行部署方案利弊权衡(calculate re-deployment plan & evaluate benefit/cost of re-depoyment).本文从用户满意度增加的角度衡量重配置获得的好处.为了防止频繁触发重配置时机,本文采用阈值判定方法来判断是否需要进行重配置,判断条件为 $\frac{satisfation(depl_{new})-satisfation(depl_{old})}{satisfation(depl_{old})}>threshold$.只有当新的部署方案对于用户满意度的提高大于阈值时,才会发生重配置.若确需重配置,服务器则将部署方案(re-deployment plan)发送给移动客户端完成重配置过程.

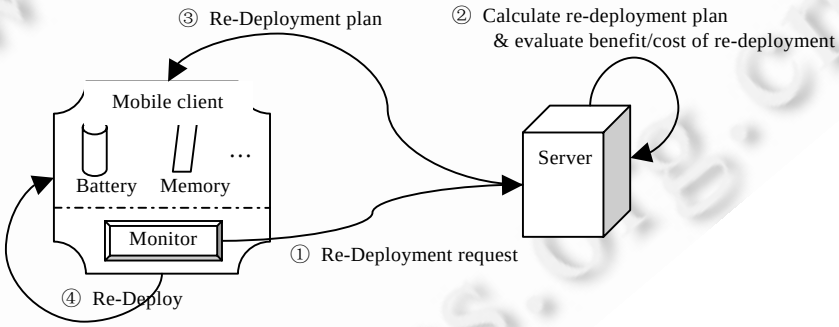


Fig.3 Re-Configuration process

图 3 部署方案重配置过程概览

另外,本文开发了基于 Prism-MW[9,10]的工具 NanoDepl 以辅助生成部署方案,其以软件体系结构规约和“个性化用户信息”为输入,生成满足用户个性化需求的部署方案.图 4 是 NanoDepl 应用于移动相册的示例情景,它可以将应用的体系结构和用户的个性化需求描述以图形化的方式加以展示,并对部署方案生成算法进行整合,从而为部署方案生成工作提供有效支持.

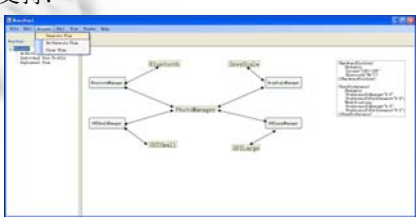


Fig.4 NanoDepl tool snapshot

图 4 NanoDepl 工具示例

2.5 应用示例研究

在“移动相册”的应用实例中,移动客户端从服务器端动态下载适应性软件以满足特定用户的需求,图 5 展示了其软件体系结构规约,该应用由 6 个构件和 6 个连接子组成,这些基本构件和连接子为满足不同的用户需求而开发.

```
⟨Component⟩
    ⟨name⟩Bluetooth⟨/name⟩
    ⟨providedinterface⟩Send⟨/provideinterface⟩
    ⟨providedinterface⟩Receive⟨/provideinterface⟩
    ⟨requiredinterface⟩Decode⟨/requiredinterface⟩
    ⟨DeviceParameter⟩
    ⟨Bluetooth⟩“Yes”⟨/Bluetooth⟩
    ⟨/DeviceParameter⟩
⟨/Component⟩
⟨Component⟩
    ⟨name⟩GreyScale⟨/name⟩
    ⟨providedinterface⟩ConvertFormat⟨/provideinterface⟩
    ⟨preferencetoenergy⟩“0.7”⟨/preferencetoenergy⟩
⟨/Component⟩
⟨Component⟩
    ⟨name⟩GUISmall⟨/name⟩
    ⟨providedinterface⟩Display⟨/provideinterface⟩
    ⟨screen⟩“0.4”⟨/screen⟩
⟨/Component⟩
⟨Component⟩
    ⟨name⟩GUILarge⟨/name⟩
    ⟨providedinterface⟩Display⟨/provideinterface⟩
    ⟨screen⟩“0.6”⟨/screen⟩
⟨/Component⟩
⟨Component⟩
    ⟨name⟩PhotoManager⟨/name⟩
    ⟨providedinterface⟩Add⟨/provideinterface⟩
    ⟨providedinterface⟩Remove⟨/provideinterface⟩
    ⟨providedinterface⟩Find⟨/provideinterface⟩
    ⟨providedinterface⟩Update⟨/provideinterface⟩
⟨/Component⟩
⟨Component⟩
    ⟨name⟩Network⟨/name⟩
    ⟨providedinterface⟩Send⟨/provideinterface⟩
    ⟨providedinterface⟩Receive⟨/provideinterface⟩
    ⟨requiredinterface⟩Decode⟨/requiredinterface⟩
⟨/Component⟩

⟨Connector⟩
    ⟨name⟩GUISmall-Manager⟨/name⟩
    ⟨connectedcomponent⟩GUISmall⟨/connectedcomponent⟩
    ⟨connectedcomponent⟩PhotoManager⟨/connectedcomponent⟩
⟨/Connector⟩
⟨Connector⟩
    ⟨name⟩GUILarge-Manager⟨/name⟩
    ⟨connectedcomponent⟩GUILarge⟨/connectedcomponent⟩
    ⟨connectedcomponent⟩PhotoManager⟨/connectedcomponent⟩
⟨/Connector⟩
⟨Connector⟩
    ⟨name⟩GreyScale-Manager⟨/name⟩
    ⟨connectedcomponent⟩GreyScale⟨/connectedcomponent⟩
    ⟨connectedcomponent⟩PhotoManager⟨/connectedcomponent⟩
⟨/Connector⟩
⟨Connector⟩
    ⟨name⟩Bluetooth-Manager⟨/name⟩
    ⟨connectedcomponent⟩Bluetooth⟨/connectedcomponent⟩
    ⟨connectedcomponent⟩PhotoManager⟨/connectedcomponent⟩
⟨/Connector⟩
⟨Connector⟩
    ⟨name⟩EventHandling⟨/name⟩
    ⟨connectedcomponent⟩GUI⟨/connectedcomponent⟩
    ⟨connectedcomponent⟩Bluetooth⟨/connectedcomponent⟩
    ⟨QoSParameter⟩
    ⟨AvgEventSize⟩“500b”⟨/AvgEventSize⟩
    ⟨/QoSParameter⟩
⟨/Connector⟩
⟨Connector⟩
    ⟨name⟩Selection⟨/name⟩
    ⟨connectedcomponent⟩Network⟨/connectedcomponent⟩
    ⟨connectedcomponent⟩GreyScale⟨/connectedcomponent⟩
⟨/Connector⟩
```

Fig.5 Architecture specification for mobile album application

图 5 移动相册应用的软件体系结构规约

基于该应用,移动用户可通过 *PhotoManager* 对照片进行本地管理,通过 *Bluetooth* 进行照片的点对点传输,通过若干 *GUI** 完成对照片的展示.若用户需要节约功耗,可通过 *GreyScale* 来达成目标.连接子 *GreyScale-Manager* 则将 *GreyScale* 和 *PhotoManager* 连接起来.

为了生成适应性部署方案,除了体系结构规约外,还需要对“个性化用户信息”进行描述.图 6 展示了“个性化用户信息”示例,其对硬件平台和用户偏好的需求进行规约.另外,如第 2.3 节所示,亦需要建立非功能属性与其语义信息的映射关系.这里,本文定义当移动设备屏幕大小小于 176×220 时为小屏幕,大于 240×320 时为大屏幕,否则为正常屏幕.接下来,建立 *cmPara*(*cm*,*cmp*),*UPAPrio*(*upa*)等函数的映射关系,如图 7 所示.

基于这些输入,利用 MINLP 算法,NanoDepl 工具生成的最终优化部署方案为

depl′={*GreyScale*,*GUISmall*,*PhotoManager*,*GUISmall-Manager*,*GreyScale-Manager*,*Network*,*Selection*}.

可以看出,该部署方案不仅符合应用部署目标平台的物理特性,也符合图 6 所示的用户个性化需求,并最大化了衡量用户满意度的函数 *satisfaction*(*d*).

```
〈HardwarePlatform〉
  〈Defaults
   Screen="128×128"
   Bluetooth="No"/〉
〈/HardwarePlatform〉
〈UserPreference〉
  〈Defaults
   PreferenceToEnergy="0.5"
   PreferenceToPerformance="0.5"/〉
  〈Modifications
   PreferenceToEnergy ="0.8"
   PreferenceToPerformance="0.2"/〉
〈/UserPreference〉
```

Fig.6 Example of user profile

图 6 "个性化用户信息"示例

$$cmPara(Bluetooth, Bluetooth) = 1$$
$$cmPara(GUISmall, screen) = 0.4$$
$$cmPara(GUILarge, screen) = 0.6$$
$$cmPara(GreyScale, preferencetoenergy) = 0.7$$
$$UPAPrio(preferencetoenergy) = 2.5$$
$$UPAPrio(screen) = 2$$
$$UPAPrio(Bluetooth) = 1.5$$
$$AC = \{Bluetooth, screen\}$$

Fig.7 Relationship example

图 7 函数映射关系示例

3 实验与分析

3.1 算法实验效果分析

算法效率对 ADCA 方法的整体执行效率具有重要影响,因此本文对两种算法的执行效果进行分别实验与测试.实验的硬件平台基于联想启天 M6900 服务器,其中,处理器型号为 Intel 奔腾 E5400,内存为 1G DDRII 800.软件平台基于 Solaris 10 Update 9[17].本文以移动 Bomber[18]作为应用示例来展示贪心算法和 MINLP 算法的执行效率.移动 Bomber 是一个典型的移动版 3D 飞行模拟游戏,其完全版本由 67 个构件/连接子组成.由于需要测试算法在不同构件数目条件下的效率,本文对移动 Bomber 进行不同程度的裁剪,使其成为不同数目构件/连接子构成的应用,并对每个剪裁版本进行算法效率测试.

在存在时间限制的条件下,本文采用贪心算法来生成部署方案,尽可能地提高用户满意度.图 8 展示了在不同的时间限制条件下,针对不同构件数目的应用,采用贪心算法获得的用户满意度提升情况.这里,用户满意度提升以优化前后的部署方案为比较基础,以用户满意度的提升百分比为量化基准,即

$$improvement\ on\ user\ satisfation = \frac{satisfation(depl_{improved}) - satisfation(depl_{original})}{satisfation(depl_{original})} \times 100\%.$$

在计算用户满意度提升的过程中,需要设定初始部署方案作为比较基础.这里,针对不同构件数目的应用,以最小构件集合作为初始部署方案,当存在多个最小构件集合时,随机选取一个作为初始部署方案.以第 3.1 节中的"移动相册"应用为例,$depl_1$={*GUISmall*,*PhotoManager*,*GUISmall-Manager*}和 $depl_2$={*GUILarge*,*PhotoManager*,*GUILarge-Manager*}均是最小构件集合,在实验过程中,本文的算法将从中任意选取一个部署方案作为比较基准.从实验结果可以看出,即使在构件数目较多的情况下,采用贪心算法亦可以提高用户满意度 30%左右,如图 8 所示.

在生成优化部署方案时,采用 MINLP 可以生成最大化 *satisfaction*(*d*)的部署方案,最大程度地改善用户满意度.但是当构件数目过多时,有可能带来算法消耗时间过多的问题.图 9 展示了在不同构件数目的条件下,采用 MINLP 算法生成最优部署方案所消耗的时间.从实验结果可以看出,当构件数目小于 20 时,MINLP 所消耗的时间与构件数目基本呈线性关系;之后,随着构件数目的增多,消耗时间增长明显变快.

从图 8 和图 9 对贪心算法和 MINLP 实验结果的展示可以看出:当构件数目较少或无时间限制时,宜采用 MINLP 算法来生成最优部署方案;而当构件数目较多或时间限制严格时,宜采用贪心算法生成部分优化的部署方案.这一选取原则以配置文件的方式存在于 NanoDepl 工具之中,配置文件标明在不同时间限制和构件数目限制的条件下所选取的算法.默认的配置规则以图 8 和图 9 的实验结果为依据,如下所示:

- 若构件数目≥20∨时间限制≤400ms,采用贪心算法;
- 否则,采用 MINLP 算法.

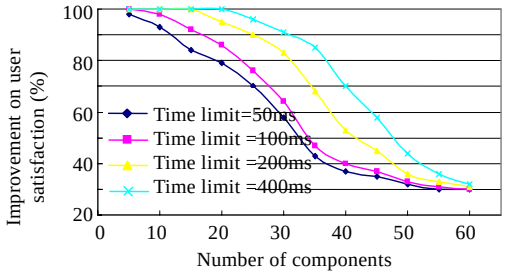


Fig.8 Effect of greedy algorithm on user satisfaction

图 8 采用贪心算法对用户满意度的提升

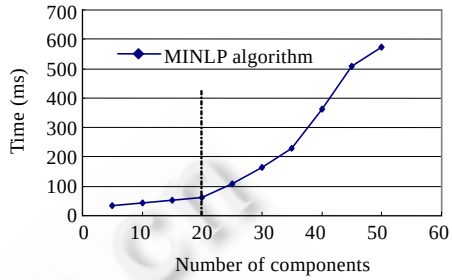


Fig.9 Time consumption of MINLP algorithm

图 9 MINLP 算法的时间开销比较

3.2 重配置实验结果分析

```
〈HardwarePlatform〉
  〈Defaults
    Screen="128×128"
    Bluetooth="No"/〉
〈/HardwarePlatform〉
〈UserPreference〉
  〈Defaults
    PreferenceToEnergy="0.5"
    PreferenceToPerformance="0.5"/〉
  〈Modifications
    PreferenceToEnergy ="0.2"
    PreferenceToPerformance="0.8"/〉
〈/UserPreference〉
```

ADCA 方法的一个特点在于对部署方案的运行时监控和重配置支持.以第 3.1 节中的"移动相册"应用为例,左侧为"个性化用户信息",当初始时,其对能耗的偏好表明在初始状态下由于电量充足,用户更偏好于提高性能.此时,NanoDepl 为用户生成的部署方案为 $depl=\{GUISmall,PhotoManager,GUISmall\text{-}Manager\}$.

随着时间的推移,用户对于节约能耗的偏好会逐渐增强,这使得原部署方案 $depl$ 对于用户偏好的满足程度不断降低.根据本文的重配置算法,当 $\frac{satisfation(depl_{new})-satisfation(depl_{old})}{satisfation(depl_{old})}>threshold$ 时将进行部署方案的重配置,新的部署方案将显著提高用户的满意程度.这里设置 $threshold$ 为 0.3,即当用户满意度的提高大于 30%时,重配置才会被触发.图 10 展示了"移动相册"应用在用户偏好 *PreferenceToEnergy* 从 0.2 开始不断增加的过程中,在启用重配置机制和不启用重配置机制两种情况下用户满意度的对比.

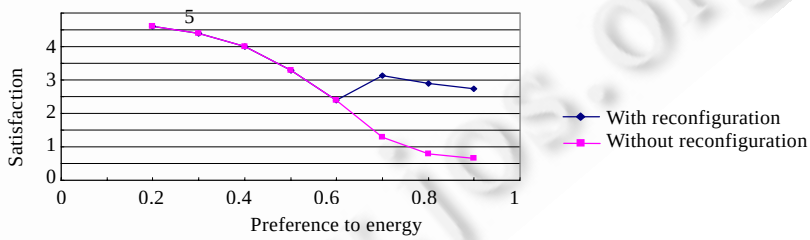


Fig.10 Reconfiguration effect of ADCA approach

图 10 ADCA 方法的重配置效果

可以看出,随着移动终端电源的不断消耗,用户节约功耗的偏好增强,造成初始配置方案与应用当前环境匹配程度的降低.当重配置时机被触发时,新的部署方案可以有效地改善用户的满意度.在该实验中,当用户偏好 *PreferenceToEnergy* 增至 0.6 时,触发重配置时机,如图 10 所示.实验结果表明,相对于无重配置机制而言,在适当的时刻进行应用重配置,可以有效改善用户的满意度.

综合以上实验结果,本文提出的自动优化部署方案的方法可以很好地满足用户的个性化需求,提高用户满意度.同时,所采用的算法组合执行效率较高,可以在规定的时限范围内获得优化的部署方案.另外,部署方案的自动重配置方法可以在应用环境不断变化的情况下有效地改善用户满意度.

4 讨论与结束语

4.1 方法适用范围讨论

本文针对目前广泛使用的手持移动设备,提出了一种体系结构驱动的、以尽可能满足用户个性化需求为目标的、用于网络化移动应用的客户端部署方案优化方法 ADCA.由于目前网络化移动应用客户端通常不是分布式的,因此在生成部署方案时,ADCA 方法尚没有考虑构件的分布式部署.

另外,在试图生成满足用户个性化需求的移动应用部署方案时,综合考量应用内部构件选择、应用系统参数配置以及运行环境参数配置对于目标达成非常重要.但由于移动应用客户端往往需要支持多种应用,在为单个移动应用生成部署方案时,针对其形成的应用系统和运行环境参数配置可能对其他应用的运行性能有所影响.移动应用客户端的构件选择和应用系统/运行环境参数配置是密切相关的两项工作.目前,ADCA 方法针对内部构件组成给出了部署方案.

4.2 结束语

本文提出的 ADCA 方法,首先对网络化移动应用的个性化特点加以分析,然后对用户个性化需求的 3 大驱动力——移动终端软/硬件平台多样性、用户偏好个人化、QoS 需求多样性——加以分析,并在 ADCA 方法中对其进行综合考量.ADCA 方法的主要步骤包括体系结构建模、属性分析和部署方案生成、运行时监控和重部署等,并将生成满足用户个性化需求的部署方案这一问题转化为整数规划、非线性规划问题加以求解.最后,通过应用实例和实验对方法的可行性及方法效果进行展示.可以看出,该方法生成的部署方案可以有效提高用户满意度.同时,运行时监控和重配置可以在应用环境或用户个性化需求发生改变时,实现对用户需求的满足.

未来工作中,希望将用户个性化需求的自动感知机制和基于用户需求自动感知的重配置时机选择机制加入到 ADCA 方法中.用户个性化需求的自动感知对于重配置机制的效果具有重要影响,因为用户需求感知结果的精确度对度量当前用户对部署方案的满意程度很重要,因此希望在未来找到合适的方法来解决精确感知用户个性化需求的问题.而基于用户需求自动感知的重配置时机选择机制对于准确把握重配置时机,更好地满足用户不断变化的个性化需求具有重要意义.这些后续工作可以为用户提供更好的个性化部署方案,进一步提高用户满意度.

另外,希望在未来工作中探讨应用构件选择和运行平台参数配置综合权衡的相关解决方案.因为运行平台的参数配置对于移动应用整体性能具有重要影响,然而针对单一移动应用生成的“运行平台参数配置”对于移动客户端运行的其他应用性能有所影响.因此,希望在未来提出合适的解决方案以全面协调.

References:

[1] Zhang XW, Cao DG, Tian G, Chen XQ. Globally adaptive data prefetching mechanism for mobile network applications. Journal of Software, 2010,21(8):1783−1794 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3617.htm [doi: 10.3724/SP.J.1001.2010.03617]

[2] Perry DE, Wolf AL. Foundations for the study of software architecture. ACM SIGSOFT Software Engineering Notes, 1992,17(4): 40−52. [doi: 10.1145/141874.141884]

[3] Ali N, Solís C, Ramos I. Comparing architecture description languages for mobile software systems. In: Proc. of the 1st Int’l Workshop on Software Architectures and Mobility. New York: ACM Press, 2008. 33−38. [doi: 10.1145/1370888.1370897]

[4] Clements P, Kazman R, Klein M. Evaluating Software Architectures: Methods and Case Studies. Addison Wesley, 2001.

[5] Malek S, Seo C, Ravula S, Petrus B, Medvidovic N. Reconceptualizing a family of heterogeneous embedded systems via explicit architectural support. In: Proc. of the 29th Int’l Conf. on Software Engineering. Washington: IEEE Computer Society, 2007. 591−601. [doi: 10.1109/ICSE.2007.69]

[6] Mikic-Rakic M, Malek S, Medvidovic N. Architecture-Driven software mobility in support of QoS requirements. In: Proc. of the 1st Int’l Workshop on Software Architectures and Mobility. New York: ACM Press, 2008. 3−8. [doi: 10.1145/1370888.1370891]

[7] http://www.nanokit.org/projects

[8] Zhang XW, Cao DG, Mei H. Improve the portability of J2ME applications: An architecture-driven approach. In: Proc. of the 3rd Int'l Conf. on Multimedia and Ubiquitous Engineering. Los Alamitos: IEEE Computer Society, 2009. 386−391. [doi: 10.1109/MUE.2009.71]

[9] Malek S, Mikic-Rakic M, Medvidovic N. A style-aware architectural middleware for resource constrained, distributed systems. IEEE Trans. on Software Engineering, 2005,31(3):256−272. [doi: 10.1109/TSE.2005.29]

[10] http://sunset.usc.edu/~softarch/Prism/

[11] http://www.w3.org/TR/NOTE-CCPP/

[12] Krause A, Smailagic A, Siewiorek DP. Context-Aware mobile computing: Learning context-dependent personal preferences from a wearable sensor array. IEEE Trans. on Mobile Computing, 2006,5(2):113−127. [doi: 10.1109/TMC.2006.18]

[13] Nascimento LM, de Almeida ES, de Lemos Meira SR. A case study in software product lines—The case of the mobile game domain. In: Proc. of the 34th Euromicro Conf. on Software Engineering and Advanced Applications. Los Alamitos: IEEE Computer Society, 2008. 43−50. [doi: 10.1109/SEAA.2008.14]

[14] http://www.gamsworld.org/minlp/solvers.htm#BARON

[15] Sahinidis NV. Optimization under Uncertainty: State-of-the-Art and opportunities. Computers and Chemical Engineering, 2004, 28(6):971−983. [doi: 10.1016/j.compchemeng.2003.09.017]

[16] Wolsey LA. Integer Programming. New York: John Wiley & Sons, 1998.

[17] http://www.oracle.com/solaris/index.html

[18] http://sourceforge.net/projects/bombers60/

附中文参考文献:

[1] 张晓薇,曹东刚,田刚,陈向群.网络化移动应用的全局适应性数据预取机制.软件学报,2010,21(8):1783−1794. http://www.jos.org.cn/1000-9825/3617.htm [doi: 10.3724/SP.J.1001.2010.03617]

张晓薇(1983－),女,黑龙江佳木斯人,博士生,主要研究领域为软件工程,嵌入式构件技术.

曹东刚(1975－),男,博士,副教授,CCF会员,主要研究领域为中间件技术,软件工程.

陈向群(1961－),女,教授,博士生导师,CCF高级会员,主要研究领域为软件工程,操作系统,嵌入式软件.

梅宏(1963－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为软件工程与软件工程环境,软件复用和软件构件技术,分布对象技术.