软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2019,30(11):3297−3312 [doi: 10.13328/j.cnki.jos.005802]

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

智能家居情境感知服务的运行时建模与执行方法*

陈　星[1,2], 黄志明[1,2], 叶心舒[1,2], 马　郓[3], 陈艺燕[1,2], 郭文忠[1,2]

[1](福州大学　数学与计算机科学学院,福建　福州　350108)

[2](福建省网络计算与智能信息处理重点实验室(福州大学),福建　福州　350108)

[3](清华大学　软件学院,北京　100084)

通讯作者: 郭文忠, E-mail: guowenzhong@fzu.edu.cn

摘　要: 随着智能家居基础设施的不断发展,智能家居逐渐进入以智能服务为特征的新时期.大量复杂、异构的智能设备相互协同,构成海量、智能、集成的智能家居应用.其中,情境感知服务根据服务对象所处情境的变化为其提供准确的服务,是智能家居应用的典型代表.目前,情境感知服务往往面向场景进行构建,其设备多样性和服务随需性给应用开发带来极大的挑战.开发者需要熟悉设备管理接口、进行接口调用和交互,同时,理解服务功能和质量需求,进行管理逻辑的编写.为了快速定制和开发情境感知服务,将知识图谱引入开发过程,提出一种智能家居情境感知服务的运行时建模与执行方法:首先,提出智能家居情境感知服务知识图谱概念模型,定义其情境中各种概念和关系;其次,提出智能家居情境感知服务知识图谱实例模型的构造与维护机制,通过运行时概念、关系实例表示情境知识;最后,提出基于知识推理的智能家居情境感知服务执行方法,通过知识推理自动执行设备功能.面向实际场景,构建智能家居原型系统.实验结果显示,该方法能够实现情境感知服务运行时建模与执行,其代码减少量超过90%.

关键词: 运行时软件体系结构;智能家居;情境感知;知识图谱;软件自适应

中图法分类号: TP311

中文引用格式: 陈星,黄志明,叶心舒,马郓,陈艺燕,郭文忠.智能家居情境感知服务的运行时建模与执行方法.软件学报,2019, 30(11):3297−3312. http://www.jos.org.cn/1000-9825/5802.htm

英文引用格式: Chen X, Huang ZM, Ye XS, Ma Y, Chen YY, Guo WZ. Approach to modeling and executing context-aware services of smart home at runtime. Ruan Jian Xue Bao/Journal of Software, 2019,30(11):3297−3312 (in Chinese). http://www.jos.org.cn/1000-9825/5802.htm

Approach to Modeling and Executing Context-aware Services of Smart Home at Runtime

CHEN Xing[1,2], HUANG Zhi-Ming[1,2], YE Xin-Shu[1,2], MA Yun[3], CHEN Yi-Yan[1,2], GUO Wen-Zhong[1,2]

[1](College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350108, China)

[2](Fujian Provincial Key Laboratory of Networking Computing and Intelligent Information Processing (Fuzhou University), Fuzhou 350108, China)

[3](School of Software, Tsinghua University, Beijing 100084, China)

Abstract: As the infrastructure supporting smart home evolves, smart home has entered a new stage featured by intelligent services. A large number of complex and heterogeneous smart devices cooperate with each other, and make up plenty of intelligent and integrated smart home applications, in which context-aware services can be regarded as typical representatives. The context-aware services aim to provide accurate services to users according to their contexts. Developers usually design and develop these services based on scenario, and face huge challenges from device and demand variations. They first have to be familiar with the APIs provided by smart devices and then build the program upon them according to functional and nonfunctional requirements of services. In order to customize and develop

* 基金项目: 国家重点研发计划(2018YFB1004800); 福建省高校杰出青年科研人才计划; 福建省引导项目(2018H0017)

Foundation item: National Key Research and Development Program of China (2018YFB1004800); Talent Program for Distinguished Young Scholars in Higher Education of Fujian Province; Guiding Project of Fujian Province (2018H0017)

收稿时间: 2018-07-16; 修改时间: 2018-09-20; 采用时间: 2018-11-15

these services more efficiently, this study proposes an approach to model and execute context-aware services at runtime, which introduces the knowledge graph into development process. First, concepts and relations of context-aware services are defined in the concept model of knowledge map. Second, runtime instances of concepts and relations in knowledge map are used to represent the knowledge of user's context. Third, knowledge reasoning based on the runtime knowledge map is implemented to perform device functions automatically. The proposed framework is evaluated on a prototype system, and the results show that the proposed approach can model and execute context-aware services at runtime and LOC (lines of code) is reduced by 90%.

Key words: runtime software architecture; smart home; context-aware; knowledge graph; software adaption

随着智能家居基础设施的不断发展,智能家居逐渐进入以智能服务为特征的新时期,智能机器人、智能穿戴设备和智能家电等大量复杂、异构的新设备接入网络,进行交互和协同,以实现智能化识别、监控等海量应用[1−3].其中,温度调节、空气调节等情境感知服务,根据服务对象所处情境的变化为其提供准确的温度控制、空气净化等服务,是智能家居应用的典型代表.目前,情境感知服务往往面向场景进行构建,主要面临两个方面的挑战.

- 一是设备的多样性:智能设备在其功能、品牌和型号等方面存在差异,往往提供不同的数据读取和功能调用方式,给设备的交互、协同带来极大复杂度.
- 二是服务的随需性:存在不同类型的服务需求,需要建立情境知识与智能设备间的关联关系,给服务的管理逻辑编写带来极大的复杂度.

智能家居情境感知服务实际上是对服务对象所处情境的各种信息进行收集、分析、决策和实施的过程.从系统实现的角度看,需要使用场景中智能设备的管理接口来获取各种信息,并根据具体的服务需求对这些信息进行分析、决策和实施.例如在环境监控服务中,通过对房间的光亮、温度和空气质量等条件进行监测,从而自动地对电灯、空调和空气净化器等智能设备进行管理.然而,目前的情境感知服务开发,往往使用 C/Java 等通用语言直接调用智能设备提供的管理接口,这种编程方式具有良好的适应性,但会带来高昂的编程开销.开发者必须熟悉不同智能设备的管理接口,才能实现它们的交互.此外,由于应用系统建立在与特定智能设备绑定的底层代码的基础上,其管理逻辑无法复用;即使管理机制类似,开发者仍需要开发多个应用系统来适应不同场景.

智能家居情境感知服务开发面临的主要问题是:其问题域与系统实现间存在着鸿沟,通过手工编码实现问题域到系统实现的映射,会带来巨大的编程复杂性.知识图谱用来描述客观世界的概念、实体、事件及其之间的关系[4−6],能够扮演系统需求与系统实现之间的桥梁,可以用来解决智能家居情境感知服务需求到实现的映射过程中,系统复杂性所带来的问题.然而,将知识图谱引入到情境感知服务的开发过程中,仍面临以下两个方面的挑战.

- 一方面,知识图谱难以表示服务对象所处情境的变化.知识图谱能够组织结构化数据,用实体和关系进行知识表示.但是情境知识表示要求知识图谱能够感知实时场景信息,而现有的知识图谱技术难以反映数据的实时变化.
- 另一方面,面向不同服务需求构建推理规则工作量大.知识图谱用来表示情境信息,并通过推理规则实现情境感知服务的自动执行.但是智能家居服务具有开放、动态、多变的特点,给推理规则的构建带来极大复杂度.

为了能够根据场景快速定制和开发智能家居情境感知服务,本文提出一种智能家居情境感知服务的运行时建模与执行方法.首先提出智能家居情境感知服务知识图谱概念模型,定义其情境中的各种概念和关系;其次,提出智能家居情境感知服务知识图谱实例模型的构造与维护机制,通过运行时概念、关系实例表示情境知识;最后,提出基于知识推理的智能家居情境感知服务执行方法,通过知识推理自动执行设备功能.面向实际场景,构建智能家居原型系统.实验结果显示,该方法能够实现情境感知服务运行时建模与执行,其代码减少量超过 90%.

本文第 1 节概述方法的整体框架.第 2 节介绍智能家居情境感知服务知识图谱概念模型.第 3 节介绍智能家居情境感知服务知识图谱实例模型的构造与维护机制.第 4 节介绍基于知识推理的智能家居情境感知服务

的执行方法.第 5 节介绍实例研究并对方法进行评估.第 6 节与相关工作进行比较.第 7 节总结全文并展望未来的工作.

1 方法概览

图 1 是智能家居情境感知服务的运行时建模与执行方法概览.方法将知识图谱引入到服务开发过程中,通过情境知识定义、情境知识表示和情境知识推理,实现情境感知服务的开发自动化.该方法主要包含 3 部分工作:1) 智能家居情境感知服务知识图谱概念模型;2) 智能家居情境感知服务知识图谱实例模型运行时建模方法;3) 基于知识推理的智能家居情境感知服务执行方法.

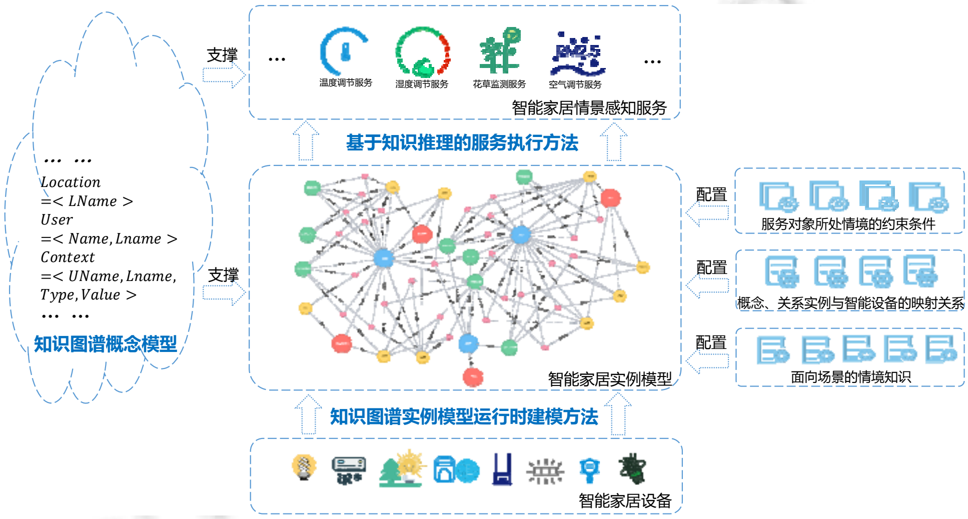

Fig.1 Overview of approach to modeling and executing context-aware services of smart home at runtime

图 1 智能家居情境感知服务的运行时建模与执行方法概览

- 首先提出智能家居情境感知服务知识图谱的概念模型.概念模型是描述智能家居场景中概念和关系等抽象元素的统一模型,定义了位置、用户、环境、设备、服务等概念以及位于、感知、提供、监测、提高、降低等关系;其实例模型是各抽象元素的具体实例,而情境感知服务的知识推理则基于概念层次的知识抽象.
- 其次,提出智能家居情境感知服务知识图谱实例模型的运行时建模方法.实例模型通过概念、关系实例表示智能家居的情境知识,描述具体场景中客观存在的事实;基于前期工作运行时软件体系结构模型[7,8],设计不同类型概念、关系实例的运行时建模方法,并建立知识图谱实例模型与具体智能家居场景的双向同步机制.
- 最后,提出基于知识推理的智能家居情境感知服务执行方法.情境感知服务执行建立在上述智能家居运行时知识图谱的基础上,通过知识推理自动执行设备功能;服务需求本质上是智能家居情境需要满足的一组条件,将其描述为运行时知识图谱的一组约束;基于概念层次的知识抽象,设计情境感知服务的知识推理方法,自动决策需要执行的设备功能.

基于上述方法,开发者仅声明面向场景的情境知识,配置概念实例与智能设备的映射关系,描述服务对象所处情境的约束条件,就能够自动构建智能家居情境感知服务,极大地降低服务开发的难度和复杂度.

2 智能家居情境感知服务知识图谱概念模型

知识图谱概念模型是智能家居情境感知服务概念层次的知识抽象,描述了智能家居场景中概念和关系等抽象元素,如图 2 所示.

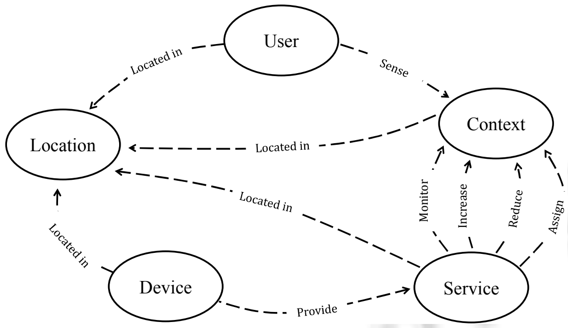

Fig.2 Concept model of knowledge map for context-aware services of smart home

图 2 智能家居情境感知服务知识图谱概念模型

概念模型定义了位置、用户、环境、设备、服务等智能家居场景中概念,见表 1.其中,位置(location)表示具体区域,属性 *LName* 表示区域名称.用户(user)表示服务对象,属性 *UName* 表示用户名称,属性 *LName* 表示用户所在区域.环境(context)表示用户敏感的某种环境状态,属性 *UName* 表示环境状态所属用户,属性 *LName* 表示环境状态所在区域,属性 *CType* 表示环境状态类型(例如光亮、温度等),属性 *CValue* 表示状态值,属性 R_{Min} 和 R_{Max} 表示状态值需要满足的范围.设备(device)表示智能设备,属性 *DName* 表示设备名称,属性 *LName* 表示设备所在区域,属性 Key_i 表示设备的配置参数或系统指标.服务(service)表示智能设备提供的监测或改变环境状态的某种服务,属性 *DName* 表示提供该服务的设备,属性 *LName* 表示服务所在区域,属性 *CType* 表示服务监控的环境状态类型,属性 *Effect* 表示服务对环境状态的作用,主要包括监测(monitor)、提高(increase)、降低(reduce)、赋值(assign)这 4 种类型.*Status* 表示服务是否开启,*SValue* 表示环境状态指标(监测)或服务配置参数(赋值).

Table 1 Concepts and their attributes in concept model of knowledge map for smart home

表 1 智能家居知识图谱概念模型中的概念及其属性

概念名	概念属性
位置(location)	$\langle Lname\rangle$
用户(user)	$\langle UName,LName\rangle$
环境(context)	$\langle UName,LName,CType,CValue,\{R_{\mathrm{Min}},R_{\mathrm{Max}}\}\rangle$
设备(device)	$\langle DName,LName,\{Key_1,Key_2,\ldots,Key_n\}\rangle$
服务(service)	$\langle DName,LName,CType,Effect,Status,SValue\rangle$

概念模型还定义了位于、感知、提供、监测、提高、降低、赋值等上述概念之间的关系,如图 2 所示.其中,位于 ($X \xrightarrow{Located\ in} C$) 表示用户 *U*、环境 *C*、设备 *D*、服务 *S* 等位于位置 *L*,感知 ($U \xrightarrow{Sense} C$) 用户 *U* 感知环境 *C* 的状态,提供 ($D \xrightarrow{Provide} S$) 表示设备 *D* 提供服务 *S*,监测 ($S \xrightarrow{Monitor} C$) 表示服务 *S* 用来监测环境 *C* 的状态值,提高 ($S \xrightarrow{Increase} C$) 表示服务 *S* 用来提高环境 *C* 的状态值,降低 ($S \xrightarrow{Reduce} C$) 表示服务 *S* 用来降低环境 *C* 的状态值,赋值 ($S \xrightarrow{Assign} C$) 表示服务 *S* 用来改变环境 *C* 的状态值.

3 智能家居情境感知服务知识图谱实例模型运行时建模方法

知识图谱实例模型通过概念、关系实例表示智能家居情境感知服务的情境知识,设计概念、关系实例的运行时建模方法,实现知识图谱实例模型与具体智能家居场景的双向同步.

3.1 概念实例的运行时建模

知识图谱概念模型定义了位置、用户、环境、设备、服务等智能家居场景中概念,基于开发者配置构造其概念实例,并通过模型操作转换实现概念实例属性与场景实时信息的双向同步.

开发者提供的相关配置包括面向场景的情境知识、概念实例与智能设备的映射关系、服务对象所处情境的约束条件,描述了场景中具体的位置、用户、环境、设备和服务等客观事物.面向场景的情境知识提供了具体场景中的位置集合$\{L_1,L_2,\ldots,L_n\}$;概念实例与智能设备的映射关系提供了具体场景中的设备集合$\{D_1,D_2,\ldots,D_n\}$、其提供服务的集合$\{S_1,S_2,\ldots,S_n\}$以及设备与服务的关联关系;服务对象所处情境的约束条件描述了具体场景中的用户集合$\{U_1,U_2,\ldots,U_n\}$、其定位装置的集合$\{T_1,T_2,\ldots,T_n\}$以及用户敏感的环境状态集合$\{C_1,C_2,\ldots,C_n\}$.于是,根据位置集合 L、用户集合 U、环境集合 C、设备集合 D 和服务集合 S,构造相应类型的概念实例.同时,本文使用 SM@RT 工具[7,8]构造智能设备和定位装置的运行时模型,并通过运行时模型实现在模型层对智能设备和定位装置进行管理[9].

概念实例的模型操作主要包括 3 种类型,分别是 *List*、*Get* 和 *Set*.其中,*List* 表示获取该类型概念的所有实例及其属性,*Get* 表示读取概念实例的属性值,*Set* 表示写入概念实例的属性值.为了维护知识图谱概念实例属性与场景实时信息的双向同步,定义概念实例的模型操作转换规则,见表 2.

Table 2 Mapping rules for model operations of concept instances

表 2 概念实例的模型操作映射规则

	Location	User	Context
List	$List{*}L\rightarrow\{L_1,L_2,\ldots,L_n\}$ $Get\ L_i.properties\rightarrow L_i.properties$	$List{*}U\rightarrow\{U_1,U_2,\ldots,U_n\}$ $Get\ U_i.properties\rightarrow U_i.properties$	$List{*}C\rightarrow\{C_1,C_2,\ldots,C_n\}$ $Get\ C_i.properties\rightarrow C_i.properties$
Get	$Get\ L_i.LName\rightarrow L_i.LName$	$Get\ U_i.UName\rightarrow U_i.UName$ $Get\ U_i.LName\rightarrow$ $\boldsymbol{RTModel}(Get\ T_i.location)$	$Get\ C_i.UName\rightarrow C_i.UName$ $Get\ C_i.LName\rightarrow Get\ U_j.LName((\exists U_j)(U_j\xrightarrow{Sense}C_i))$ $Get\ C_i.CType\rightarrow C_i.Ctype$ $Get\ C_i.CValue\rightarrow$ $Get\ S_j.SValue((\exists S_j)((S_j\xrightarrow{Monitor}C_i)\wedge(S_j.Status=\text{"On"})))$ $Get\ C_i.R_{\mathrm{Min}}\rightarrow C_i.R\mathrm{min}$ $Get\ C_i.R_{\mathrm{Max}}\rightarrow C_i.R_{\mathrm{Max}}$
Set	–	–	–

	Device	Service
List	$List{*}D\rightarrow\{D_1,D_2,\ldots,D_n\}$ $Get\ D_i.properties\rightarrow D_i.properties$	$List{*}S\rightarrow\{S_1,S_2,\ldots,S_n\}$ $Get\ S_i.properties\rightarrow S_i.properties$
Get	$Get\ D_i.DName\rightarrow D_i.DName$ $Get\ D_i.LName\rightarrow D_i.LName$ $Get\ D_i.key_m\rightarrow\boldsymbol{RTModel}(Get\ D_i.key_m)$	$Get\ S_i.DName\rightarrow S_i.DName$ $Get\ S_i.LName\rightarrow Get\ D_j.LName((\exists D_j)(D_j\xrightarrow{Provide}S_i))$ $Get\ S_i.CType\rightarrow S_i.CType$ $Get\ S_i.Effect\rightarrow S_i.Effect$ $Get\ S_i.Status\rightarrow Get\ D_j.Key_m((\exists D_j)(D_j\xrightarrow{Provide}S_i))$ $Get\ S_i.SValue\rightarrow Get\ D_j.Key_n((\exists D_j)(D_j\xrightarrow{Provide}S_i))$
Set	$Set\ D_i.key_m\rightarrow\boldsymbol{RTModel}(Set\ D_i.key_m)$	$Set\ S_i.Status\rightarrow Set\ D_j.Key_m((\exists D_j)(D_j\xrightarrow{Provide}S_i))$ $Set\ S_i.SValue\rightarrow Set\ D_j.Key_n((\exists D_j)(D_j\xrightarrow{Provide}S_i)\wedge(S_i.Effect=\text{"Assign"}))$

- 位置实例 L_i,其属性 *LName* 的值来自配置信息.
- 用户实例 U_i,其属性 *UName* 的值来自配置信息;其属性 *LName*,与定位装置运行时模型中对应元素 T_i 的属性 *LName* 保持值的一致,当读取 U_i 属性 *LName* 的值时,自动读取 T_i 的属性 *LName* 的值,即

$$Get\ U_i.LName\rightarrow\boldsymbol{RTModel}(Get\ T_i.location).$$

- 环境实例 C_i,其属性 *Uname*,*CType* 的值来自配置信息;其属性 *LName*,与对应用户实例 $U_j(U_j\xrightarrow{Sense}C_i)$的属性 *LName* 保持值的一致,当读取 C_i 属性 *LName* 的值时,自动读取 U_j 属性 *LName* 的值,即 $Get\ C_i.LName\rightarrow Get\ U_j.LName((\exists U_j)(U_j\xrightarrow{Sense}C_i))$;其属性 *CValue*,与对应服务实例 $S_j(S_j\xrightarrow{Monitor}$

C_i)的属性 *SValue* 保持值的一致,当读取 C_i 属性 *CValue* 的值时,自动读取 S_j 属性 *SValue* 的值,即 $Get\ C_i.CValue \rightarrow Get\ S_j.SValue((\exists S_j)(S_j \xrightarrow{Monitor} C_i))$;其属性 R_{Min} 和 R_{Max} 来自配置信息.

- 设备实例 D_i,其属性 *Dname*,*LName* 的值来自配置信息;其属性 Key_m,与智能设备运行时模型中对应元素 D_i 的属性 Key_m 保持值的一致,当读取/写入 D_i 属性 Key_m 的值时,自动读取/写入运行时模型中对应元素 D_i 属性 Key_m 的值,即 $Get\ D_i.key_m \rightarrow \boldsymbol{RTModel}(Get\ D_i.key_m)/Set\ D_i.key_m \rightarrow \boldsymbol{RTModel}(Set\ D_i.key_m)$.
- 服务实例 S_i,其属性 *Dname*,*Ctype*,*Effect* 的值来自配置信息;其属性 *LName*,与对应设备实例 D_j ($D_j \xrightarrow{Provide} S_i$)的属性 *LName* 保持值的一致,当读取 S_i 属性 *LName* 的值时,自动读取 D_j 属性 *LName* 的值,即 $Get\ S_i.LName \rightarrow Get\ D_j.LName((\exists D_j)(D_j \xrightarrow{Provide} S_i))$;其属性 *Status*,与对应设备实例 D_j 中表示设备是否开启的属性 Key_m 保持值的一致,当读取/写入 S_i 属性 *Status* 的值时,自动读取/写入 D_i 属性 Key_m 的值,即 $Get\ S_i.Status \rightarrow Get\ D_j.Key_m\ /\ Set\ S_i.Status \rightarrow Set\ D_j.Key_m((\exists D_j)(D_j \xrightarrow{Provide} S_i))$;其属性 *SValue*,与对应设备实例 D_j 中表示对应环境状态的属性 Key_n 保持值的一致,当读取/写入 S_i 属性 *SValue* 的值时,自动读取/写入 D_i 属性 Key_n 的值,即

$$Get\ S_i.SValue \rightarrow Get\ D_j.Key_n\ /\ Set\ S_i.SValue \rightarrow Set\ D_j.Key_n((\exists D_j)(D_j \xrightarrow{Provide} S_i)).$$

3.2 关系实例的运行时建模

知识图谱概念模型定义了位于、感知、提供、监测、提高、降低、赋值等智能家居场景中概念之间的关系,进一步定义关系实例的运行时构造规则,见表 3.当两个概念实例满足特定前置条件时,即构造它们之间的关系实例.其中,存在用户、环境、设备、服务等类型的实例 X_i 与位置实例 L_j,X_i 属性 *LName* 的值与 L_j 属性 *LName* 的值相同时,构造关系实例 $X_i \xrightarrow{Located\ in} C_j$,表示概念实例 X_i 位于位置实例 L_j.关系实例 $U_i \xrightarrow{Sense} C_j$ 表示用户实例 U_i 感知环境实例 C_j,关系实例 $D_i \xrightarrow{Provide} S_j$ 表示设备实例 D_i 提供服务实例 S_j 均来自配置信息.存在服务实例 S_i 与环境实例 C_j,S_i 属性 *Lname*,*CType* 的值与 C_j 属性 *Lname*,*CType* 的值均相同,且 S_i 属性 *Effect* 的值为 *Monitor* 时,构造关系实例 $S_i \xrightarrow{Monitor} C_j$,表示服务实例 S_i 用来监测环境实例 C_j 的状态值.存在服务实例 S_i 与环境实例 C_j,S_i 属性 *Lname*,*CType* 的值与 C_j 属性 *Lname*,*CType* 的值均相同,且 S_i 属性 *Effect* 的值为 Increase 时,构造关系实例 $S_i \xrightarrow{Increase} C_j$,表示服务实例 S_i 用来提高环境实例 C_j 的状态值.存在服务实例 S_i 与环境实例 C_j,S_i 属性 *Lname*,*CType* 的值与 C_j 属性 *Lname*,*CType* 的值均相同,且 S_i 属性 *Effect* 的值为 Reduce 时,构造关系实例 $S_i \xrightarrow{Reduce} C_j$,表示服务实例 S_i 用来降低环境实例 C_j 的状态值.存在服务实例 S_i 与环境实例 C_j,S_i 属性 *Lname*,*CType* 的值与 C_j 属性 *Lname*,*CType* 的值均相同,且 S_i 属性 *Effect* 的值为 Assign 时,构造关系实例 $S_i \xrightarrow{Assign} C_j$,表示服务实例 S_i 用来改变环境实例 C_j 的状态值.此外,由于概念实例属性与场景实时信息保持双向同步,其关系实例将随其属性值变化而发生改变.

Table 3 Rules for constructing relation instances

表 3 关系实例的构造规则

关系实例	前置条件
$X_i \xrightarrow{Located\ in} C_j$	$X_i.LName=L_j.LName$
$U_i \xrightarrow{Sense} C_j$	–
$D_i \xrightarrow{Provide} S_j$	–
$S_i \xrightarrow{Monitor} C_j$	$S_i.LName=C_j.LName \wedge S_i.CType=C_j.CType \wedge S_i.Effect=$“*Monitor*”
$S_i \xrightarrow{Increase} C_j$	$S_i.LName=C_j.LName \wedge S_i.CType=C_j.CType \wedge S_i.Effect=$“*Increase*”
$S_i \xrightarrow{Reduce} C_j$	$S_i.LName=C_j.LName \wedge S_i.CType=C_j.CType \wedge S_i.Effect=$“*Reduce*”
$S_i \xrightarrow{Assign} C_j$	$S_i.LName=C_j.LName \wedge S_i.CType=C_j.CType \wedge S_i.Effect=$“*Assign*”

3.3 实例模型的状态更新

为了维护智能家居知识图谱实例模型与场景实时信息的双向同步,实例模型自动持续更新状态.

(1) 更新概念实例,依次为位置实例、用户实例、设备实例、服务实例和环境实例.

(2) 更新关系实例,依次为“位于”实例、“监测”实例、“提高”实例、“降低”实例和“赋值”实例.

(3) 重复执行步骤(1).

4 基于知识推理的智能家居情境感知服务执行方法

智能家居情境感知服务的执行,建立在上述智能家居知识图谱实例模型的基础上,面向服务对象所处情境的约束条件,通过知识推理,自动决策需要执行的设备功能.

情境感知服务需求本质上是服务对象所处情境的一些约束条件,即服务对象敏感的某些环境状态.在智能家居知识图谱实例模型中,用户实例 U_i 表示服务对象;环境实例 C_j 表示用户敏感的环境状态 $U_i \xrightarrow{Sense} C_j$;环境实例 C_j 的属性 *CType* 表示状态类型;属性 *CValue* 表示状态值;属性 R_{Min} 和 R_{Max} 表示其状态值需要满足的范围,即情境感知服务的服务需求.同时,设备实例 D_i 表示智能设备;服务实例 S_j 表示设备提供的功能 $D_i \xrightarrow{Provide} S_j$;服务实例 S_j 的属性 *CType* 表示监控的状态类型;属性 *Effect* 表示对状态值的影响,主要包括 Monitor、Increase、Reduce、Assign 这 4 种类型,即情境感知服务的设备功能.于是,情境感知服务的执行可以表示为:在智能家居知识图谱实例模型基础上,根据环境实例的状态值及其约束来决策相关服务实例是否开启的过程.

为了实现上述决策过程的自动化,针对不同类型的设备服务,提出通用的知识推理规则,见表 4.

Table 4 Rules for knowledge reasoning of context-aware services

表 4 情境感知服务的知识推理规则

		规则描述
Rule 1	1-1	$(\exists S_i)(\exists C_j)(S_i \xrightarrow{Monitor} C_j) \Rightarrow S_i.Status = \text{On}$
	1-2	$(\exists S_i)((S_i.Effect = \text{"}Monitor\text{"}) \wedge (\forall C_j)(\nexists S_i \xrightarrow{Monitor} C_j)) \Rightarrow S_i.Status = \text{Off}$
Rule 2	2-1	$(\exists S_i)(\exists C_j)((S_i \xrightarrow{Increase} C_j) \wedge (C_j.CValue < C_j.R_{\text{Min}})) \Rightarrow S_i.Status = \text{On}$
	2-2	$(\exists S_i)(\exists C_j)((S_i \xrightarrow{Increase} C_j) \wedge (C_j.CValue > (C_j.R_{\text{Min}} + C_j.R_{\text{Max}})/2)) \Rightarrow S_i.Status = \text{Off}$
	2-3	$(\exists S_i)((S_i.Effect = \text{"}Increase\text{"}) \wedge (\forall C_j)(\nexists S_i \xrightarrow{Increase} C_j)) \Rightarrow S_i.Status = \text{Off}$
Rule 3	3-1	$(\exists S_i)(\exists C_j)((S_i \xrightarrow{Reduce} C_j) \wedge (C_j.CValue > C_j.R_{\text{Max}})) \Rightarrow S_i.Status = \text{On}$
	3-2	$(\exists S_i)(\exists C_j)((S_i \xrightarrow{Reduce} C_j) \wedge (C_j.CValue < (C_j.R_{\text{Min}} + C_j.R_{\text{Max}})/2)) \Rightarrow S_i.Status = \text{Off}$
	3-3	$(\exists S_i)((S_i.Effect = \text{"}Reduce\text{"}) \wedge (\forall C_j)(\nexists S_i \xrightarrow{Reduce} C_j)) \Rightarrow S_i.Status = \text{Off}$
Rule 4	4-1	$(\exists S_i)(\exists C_j)((S_i \xrightarrow{Assign} C_j) \wedge (C_j.CValue > C_j.R_{\text{Max}})) \Rightarrow S_i.Status = \text{On}, S_i.SValue = (C_j.R_{\text{Min}} + C_j.R_{\text{Max}})/2$
	4-2	$(\exists S_i)(\exists C_j)((S_i \xrightarrow{Assign} C_j) \wedge (C_j.CValue < C_j.R_{\text{Min}})) \Rightarrow S_i.Status = \text{On}, S_i.SValue = (C_j.R_{\text{Min}} + C_j.R_{\text{Max}})/2$
	4-3	$(\exists S_i)((S_i.Effect = \text{"}Assign\text{"}) \wedge (\forall C_j)(\nexists S_i \xrightarrow{Assign} C_j)) \Rightarrow S_i.Status = \text{Off}$

第 1 组规则描述 Monitor 服务是否开启的推理过程:规则 1-1,存在服务实例 S_i 和环境实例 C_j 以及关系实例 $S_i \xrightarrow{Monitor} C_j$,则 S_i 属性 *Status* 的值为 On;规则 1-2,存在服务 S_i 且属性 *Effect* 的值为 Monitor,对于任一状态 C_j,不存在关系实例 $S_i \xrightarrow{Monitor} C_j$,则 S_i 属性 *Status* 的值为 Off.第 2 组规则描述 Increase 服务是否开启的推理过程:规则 2-1,存在服务实例 S_i 和环境实例 C_j 以及关系实例 $S_i \xrightarrow{Increase} C_j$,且环境状态值 $C_j.CValue$ 小于其需要满足的范围的下限 $C_j.R_{\text{Min}}$ 时,则 S_i 属性 *Status* 的值为 On;规则 2-2,存在服务实例 S_i 和环境实例 C_j 以及关系实例 $S_i \xrightarrow{Increase} C_j$,且环境状态值 $C_j.CValue$ 大于其需要满足的范围的中值($C_j.R_{\text{Min}}+C_j.R_{\text{Max}}$)/2 时,则 S_i 属性 *Status* 的值为 Off;规则 2-3,存在服务 S_i 且属性 *Effect* 的值为 Increase,对于任一状态 C_j,不存在关系实例 $S_i \xrightarrow{Increase} C_j$,则 S_i 属性 *Status* 的值为 Off.第 3 组规则描述 Reduce 服务是否开启的推理过程:规则 3-1,存在服务实例 S_i 和环

境实例 C_j 以及关系实例 $S_i \xrightarrow{Reduce} C_j$,且环境状态值 $C_j.CValue$ 大于 $C_j.R_{Max}$ 时,则 S_i 属性 *Status* 的值为 On;规则 3-2,存在服务实例 S_i 和环境实例 C_j 以及关系实例 $S_i \xrightarrow{Reduce} C_j$,且环境状态值 $C_j.CValue$ 小于($C_j.R_{Min}$+$C_j.R_{Max}$)/2 时,则 S_i 属性 *Status* 的值为 Off;规则 3-3,存在服务 S_i 且属性 *Effect* 的值为 Reduce,对于任一状态 C_j,不存在关系实例 $S_i \xrightarrow{Reduce} C_j$,则 S_i 属性 *Status* 的值为 Off.第 4 组规则描述 Assign 服务是否开启的推理过程:规则 4-1,存在服务实例 S_i 和环境实例 C_j 以及关系实例 $S_i \xrightarrow{Assign} C_j$,且环境状态值 $C_j.CValue$ 大于 $C_j.R_{Max}$ 时,则 S_i 属性 *Status* 的值为 On 且属性 *SValue* 赋值为($C_j.R_{Min}$+$C_j.R_{Max}$)/2;规则 4-2,存在服务实例 S_i 和环境实例 C_j 以及关系实例 $S_i \xrightarrow{Assign} C_j$,且环境状态值 $C_j.CValue$ 小于 $C_j.R_{Min}$ 时,则 S_i 属性 *Status* 的值为 On 且属性 *Svalue* 赋值为($C_j.R_{Min}$+$C_j.R_{Max}$)/2;规则 4-3,存在服务 S_i 且属性 *Effect* 的值为 Assign,对于任一状态 C_j,不存在关系实例 $S_i \xrightarrow{Assign} C_j$,则 S_i 属性 *Status* 的值为 Off.

5 实例研究

面向实际场景,构建智能家居原型系统,对方法进行评估.首先,验证方法是否能够实现智能家居情境感知服务的运行时建模与执行;其次,与传统方法进行对比,比较服务开发难度和执行性能.

5.1 智能家居场景

图 3 所示为智能家居场景示例.

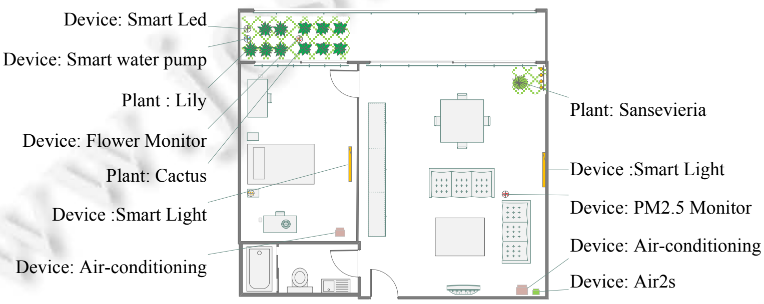

Fig.3 Example scenario of smart home

图 3 智能家居场景示例

场景包括 3 个区域,分别是客厅、卧室和阳台.各区域布置了不同的智能设备,见表 5.例如,客厅布置了智能空调、智能灯管、PM2.5 监测仪和空气净化器等设备.场景包括 4 种类型的环境状态,分别是温度、湿度、光强和 PM2.5.智能设备能够针对不同类型的环境状态,提供监测(monitor)、提高(increase)、降低(reduce)和赋值(assign)等服务,见表 6.例如,智能空调能够提供温度的监测、提高、降低和赋值等服务,空气净化器能够提供 PM2.5 的降低服务.

场景包括 5 个服务对象,分别是家庭成员 Jack 和人 Ken 以及盆栽虎尾兰、仙人掌和百合花,各服务对象位于不同区域,且具有不同的敏感环境状态,见表 7.例如,Jack 和 Ben 能够在不同区域间移动,要求温度高于 19°C 且低于 26°C,光强高于 20%,PM2.5 低于 35μg/m^3;虎尾兰布置在客厅,要求温度高于 10°C 且低于 35°C,光强高于 20%;仙人掌布置在阳台,要求湿度高于 20%,光强高于 90%;百合花布置在阳台,要求湿度高于 60%,光强高于 70%.

Table 5 Smart devices in each locations

表 5 各区域布置的智能设备

	客厅(sitting room)	卧室(bedroom)	阳台(balcony)
智能空调(air-conditioning)	1	1	–
智能灯管(smart light)	1	1	–
PM2.5 监测仪(PM2.5 monitor)	1	–	–
空气净化器(air2s)	1	–	–
花草检测仪(flower monitor)	–	–	1
智能水阀(smart water pump)	–	–	1
智能 LED(smart LED)	–	–	1

Table 6 Services provided by each smart devices

表 6 各智能设备提供的服务

	温度(temperature)	湿度(humidity)	光强(brightness)	PM2.5(particulate matter 2.5)
智能空调	Monitor/Increase/Reduce/Assign	–	–	–
智能灯管	–	–	Monitor/Increase/Assign	–
PM2.5 监测仪	–	–	–	Monitor
空气净化器	–	–	–	Reduce
花草检测仪	–	Monitor	Monitor	–
智能水阀	–	Increase	–	–
智能 LED	–	–	Increase/Assign	–

Table 7 Users' locations and requirements

表 7 各用户所在区域及其需求

	Jack(移动)	Ken(移动)	虎尾兰 Sansevieria(客厅)	仙人掌 Cactus(阳台)	百合花 Lily(阳台)
温度	[19°C,26°C]	[22°C,26°C]	[10°C,35°C]	–	–
湿度	–	–	–	>20%	>60%
光强	>20%	>20%	>20%	>80%	>70%
PM2.5	<35μg/m^3	<35μg/m^3	–	–	–

5.2 智能家居情境感知服务的运行时建模与执行

5.2.1 情境感知服务的运行时建模

根据上述智能家居场景知识,构造知识图谱实例模型,并维护其与场景实时信息的双向同步.

首先,根据场景知识构造知识图谱概念实例,见表 8.其中,

- 存在 3 个位置实例,分别对应场景中的 3 个区域.
- 存在 5 个用户实例,分别对应场景中的 5 个服务对象:U_1 和 U_2 分别对应 Jack 和 Ken,其属性 *LName* 表示所在区域,与定位装置运行时模型中对应的元素属性保持值的一致;U_3,U_4 和 U_5 分别对应虎尾兰、仙人掌和百合花.
- 存在 12 个环境实例,分别对应场景中各服务对象的敏感环境状态.例如,Jack(U_1)的敏感环境状态 C_{11},其所在区域 *LName* 与 U_1 的属性 *LName* 保持值的一致;环境状态类型为温度,状态值与对应服务实例的属性 *SValue* 保持值的一致,状态值的约束范围是[19°C,26°C].
- 存在 9 个设备实例,分别对应场景中各智能设备.例如,位于客厅的 PM2.5 检测仪(D_3),其属性 *LName* 表示所在区域,其属性 On_Off 和 PM2.5 分别与智能设备运行时模型中对应元素 D_3 的属性 On_Off 和 PM2.5,保持值的一致.
- 存在 21 个服务实例,分别对应场景中各智能设备提供的服务.例如,位于阳台的花草检测仪设备(D_7)提供的服务 S_{71},其所在区域 *LName* 与 D_7 的属性 *LName* 保持值的一致;环境状态类型为湿度(*humidity*);服务类型为监测(*monitor*),其属性 *Status* 与对应设备实例 D_7 的属性 On_Off 保持值的一致,其属性 *SValue* 与对应设备实例 D_7 的属性 *Humidity* 保持值的一致.

Table 8 Concept instances in the knowledge graph for smart home

表 8 智能家居知识图谱概念实例

概念		概念实例
位置	(location)	L_1:⟨*Lname*:sitting room⟩,L_2:⟨*Lname*:bedroom⟩,L_3:⟨*Lname*:balcony⟩
用户	(user)	U_1:⟨*UName*:Jack,*LName*:?⟩,U_2:⟨*UName*:Ken,*LName*:?⟩, U_3:⟨*UName*:Sansevieria,*LName*:sitting room⟩,U_4:⟨*UName*:Cactus,*LName*:balcony⟩, U_5:⟨*UName*:Lily,*LName*:balcony⟩
环境	(context)	C_{11}:⟨*UName*:Jack,*LName*:?,*CType*:Temperature,*CValue*:?,{R_{Min}:19°C,R_{Max}:26°C}⟩, C_{13}:⟨*UName*:Jack,*LName*:?,*CType*:Brightness,*CValue*:?,{R_{Min}:20%,R_{Max}:*}⟩, C_{14}:⟨*UName*:Jack,*LName*:?,*CType*:PM2.5,*CValue*:?,R_{Min}:*,{R_{Max}:35μg/m^3}⟩, C_{21}:⟨*UName*:Ken,*LName*:?,*CType*:Temperature,*CValue*:?,{R_{Min}:22°C,R_{Max}:26°C}⟩, C_{23}:⟨*UName*:Ken,*LName*:?,*CType*:Brightness,*CValue*:?,{R_{Min}:20%,R_{Max}:*}⟩, C_{24}:⟨*UName*:Ken,*LName*:?,*CType*:PM2.5,*CValue*:?,R_{Min}:*,{R_{Max}:35μg/m^3}⟩, C_{31}:⟨*UName*:Sansevieria,*LName*:?,*CType*:Temperature,*CValue*:?,{R_{Min}:10°C,R_{Max}:35°C}⟩, C_{33}:⟨*UName*:Sansevieria,*LName*:?,*CType*:Brightness,*CValue*:?,{R_{Min}:20%,R_{Max}:*}⟩, C_{42}:⟨*UName*:Cactus,*LName*:?,*CType*:Humidity,*CValue*:?,{R_{Min}:20%,R_{Max}:*}⟩, C_{43}:⟨*UName*:Cactus,*LName*:?,*CType*:Brightness,*CValue*:?,{R_{Min}:80%,R_{Max}:*}⟩ C_{52}:⟨*UName*:Lily,*LName*:?,*CType*:Humidity,*CValue*:?,{R_{Min}:60%,R_{Max}:*}⟩, C_{53}:⟨*UName*:Lily,*LName*:?,*CType*:Brightness,*CValue*:?,{R_{Min}:70%,R_{Max}:*}⟩
设备	(device)	D_1:⟨*DName*:*Air-conditioning*,*LName*:sitting room,{Key_1(On_Off):?,Key_2(Temperature):?}⟩, D_2:⟨*DName*:*SmartLight*,*LName*:sitting room,{Key_1(On_Off):?,Key_2(Brightness):?}⟩, D_3:⟨*DName*:PM2.5 Monitor,*LName*:sitting room,{Key_1(On_Off):?,Key_2(PM2.5):?}⟩, D_4:⟨*DName*:Air2s,*LName*:sitting room,{Key_1(On_Off):?,Key_2(PM2.5):?}⟩, D_5:⟨*DName*:*Air-conditioning*,*LName*:bedroom,{Key_1(On_Off):?,Key_2(Temperature):?}⟩, D_6:⟨*DName*:*SmartLight*,*LName*:bedroom,{Key_1(On_Off):?,Key_2(Brightness):?}⟩, D_7:⟨*DName*:*FlowerMonitor*,*LName*:balcony,{Key_1(On_Off):?,Key_2(Humidity):?,Key_3(Brightness):?}⟩, D_8:⟨*DName*:*SmartWaterPump*,*LName*:balcony,{Key_1(On_Off):?}⟩, D_9:⟨*DName*:*SmartLed*,*LName*:balcony,{Key_1(On_Off):?,Key_2(Brightness):?}⟩
服务	(service)	S_{11}:⟨*DName*:*Air-conditioning*,*LName*:?,*CType*:Temperature,*Effect*:Monitor,*Status*:?,*SValue*:?⟩ S_{12}:⟨*DName*:*Air-conditioning*,*LName*:?,*CType*:Temperature,*Effect*:Increase,*Status*:?,*SValue*:?⟩ S_{13}:⟨*DName*:*Air-conditioning*,*LName*:?,*CType*:Temperature,*Effect*:Reduce,*Status*:?,*SValue*:?⟩ S_{14}:⟨*DName*:*Air-conditioning*,*LName*:?,*CType*:Temperature,*Effect*:Assign,*Status*:?,*SValue*:?⟩ S_{21}:⟨*DName*:*SmartLight*,*LName*:?,*CType*:Brightness,*Effect*:Monitor,*Status*:?,*SValue*:?⟩ S_{22}:⟨*DName*:*SmartLight*,*LName*:?,*CType*:Brightness,*Effect*:Increase,*Status*:?,*SValue*:?⟩ S_{23}:⟨*DName*:*SmartLight*,*LName*:?,*CType*:Brightness,*Effect*:Assign,*Status*:?,*SValue*:?⟩ S_{31}:⟨*DName*:*PMmonitor*,*LName*:?,*CType*:PM2.5,*Effect*:Monitor,*Status*:?,*SValue*:?⟩ S_{41}:⟨*DName*:Air2s, *LName*:?,*CType*:PM2.5,*Effect*:Reduce,*Status*:?,*SValue*:?,}⟩ S_{51}:⟨*DName*:*Air-conditioning*,*LName*:?,*CType*:Temperature,*Effect*:Monitor,*Status*:?,*SValue*:?⟩ S_{52}:⟨*DName*:*Air-conditioning*,*LName*:?,*CType*:Temperature,*Effect*:Increase,*Status*:?,*SValue*:?⟩ S_{53}:⟨*DName*:*Air-conditioning*,*LName*:?,*CType*:Temperature,*Effect*:Reduce,*Status*:?,*SValue*:?⟩ S_{54}:⟨*DName*:*Air-conditioning*,*LName*:?,*CType*:Temperature,*Effect*:Assign,*Status*:?,*SValue*:?⟩ S_{61}:⟨*DName*:*SmartLight*,*LName*:?,*CType*:Brightness,*Effect*:Monitor,*Status*:?,*SValue*:?⟩ S_{62}:⟨*DName*:*SmartLight*,*LName*:?,*CType*:Brightness,*Effect*:Increase,*Status*:?,*SValue*:?⟩ S_{63}:⟨*DName*:*SmartLight*,*LName*:?,*CType*:Brightness,*Effect*:Assign,*Status*:?,*SValue*:?⟩ S_{71}:⟨*DName*:*FlowerMonitor*,*LName*:?,*CType*:Humidity,*Effect*:Monitor,*Status*:?,*SValue*:?⟩ S_{72}:⟨*DName*:*FlowerMonitor*,*LName*:?,*CType*:Brightness,*Effect*:Monitor,*Status*:?,*SValue*:?⟩ S_{81}:⟨*DName*:*SmartWaterPump*,*LName*:?,*CType*:Humidity,*Effect*:Increase,*Status*:?,*SValue*:?⟩ S_{91}:⟨*DName*:*SmartLed*, *LName*:?,*CType*:Brightness,*Effect*:Increase,*Status*:?,*SValue*:?⟩ S_{92}:⟨*DName*:*SmartLed*, *LName*:?,*CType*:Brightness,*Effect*:Assign,*Status*:?,*SValue*:?⟩

其次,根据场景知识构造知识图谱关系实例,见表 9.

此时,Ken 位于客厅,而盆栽分别位于相应区域.其中,

- 存在 43 个“位于”实例,对应场景中各事物位于不同区域.例如,D_1 属性 *LName* 的值为 Sitting Room,则
$$D_1 \xrightarrow{Located\ in} L_1.$$
- 存在 12 个“感知”实例,对应场景中各服务对象敏感不同环境状态.例如,U_1 敏感的环境状态 C_1,则
$$U_1 \xrightarrow{Sense} C_1.$$
- 存在 21 个“提供”实例,对应场景中各设备对象提供不同服务.例如,D_1 提供服务 S_{11},则 $D_1 \xrightarrow{Provide} S_{11}$.
- 存在 9 个“监测”实例,对应场景中各服务对象能够监测相应环境状态值.例如,S_{11} 的属性 *Lname*,*Ctype*

与 C_{21},C_{31} 对应属性的值相同,且 S_{11} 属性 *Effect* 的值为 Monitor,则 $S_{11}\xrightarrow{Monitor}C_{21}$,$S_{11}\xrightarrow{Monitor}C_{31}$.

- 存在 8 个"提高"实例,对应场景中各服务对象能够提高相应环境状态值.例如,S_{81} 的属性 *Lname*,*Ctype* 与 C_{52} 对应属性的值相同,且 S_{81} 属性 *Effect* 的值为 Increase,则 $S_{81}\xrightarrow{Increase}C_{51}$.
- 存在 3 个"降低"实例,对应场景中各服务对象能够降低相应环境状态值.例如,S_{13} 的属性 *Lname*,*Ctype* 与 C_{21},C_{31} 对应属性的值相同,且 S_{13} 属性 *Effect* 的值为 Reduce,则 $S_{13}\xrightarrow{Reduce}C_{21}$,$S_{13}\xrightarrow{Reduce}C_{31}$.
- 存在 6 个"赋值"实例,对应场景中各服务对象能够改变相应环境状态值.例如,S_{23} 的属性 *Lname*,*Ctype* 与 C_{23},C_{33} 对应属性的值相同,且 S_{23} 属性 *Effect* 的值为 Assign,则 $S_{23}\xrightarrow{Assign}C_{23}$,$S_{23}\xrightarrow{Assign}C_{33}$.

Table 9 Relation instances in the knowledge graph for smart home

表 9 智能家居知识图谱关系实例

关系	关系实例
$X_i\xrightarrow{Located\ in}C_j$	$\langle U_2,L_1\rangle,\langle C_4,L_1\rangle,\langle D_7,L_1\rangle,\langle S_{11},L_1\rangle,\langle S_{12},L_1\rangle,\ldots$
$U_i\xrightarrow{Sense}C_j$	$\langle U_1,C_{11}\rangle,\langle U_1,C_{13}\rangle,\langle U_1,C_{14}\rangle,\langle U_2,C_{21}\rangle,\langle U_2,C_{23}\rangle,\ldots$
$D_i\xrightarrow{Provide}S_j$	$\langle D_1,S_{11}\rangle,\langle D_1,S_{12}\rangle,\langle D_1,S_{13}\rangle,\langle D_1,S_{14}\rangle,\langle D_2,S_{21}\rangle,\ldots$
$S_i\xrightarrow{Monitor}C_j$	$\langle S_{11},C_{21}\rangle,\langle S_{11},C_{31}\rangle,\langle S_{31},C_{24}\rangle,\langle S_{71},C_{42}\rangle,\langle S_{71},C_{52}\rangle,\ldots$
$S_i\xrightarrow{Increase}C_j$	$\langle S_{81},C_{42}\rangle,\langle S_{81},C_{52}\rangle,\langle S_{91},C_{43}\rangle,\langle S_{91},C_{53}\rangle,\ldots$
$S_i\xrightarrow{Reduce}C_j$	$\langle S_{13},C_{21}\rangle,\langle S_{13},C_{31}\rangle,\ldots$
$S_i\xrightarrow{Assign}C_j$	$\langle S_{14},C_{21}\rangle,\langle S_{14},C_{31}\rangle,\langle S_{23},C_{23}\rangle,\langle S_{23},C_{33}\rangle,\ldots$

最后,根据上述概念实例和关系实例构造知识图谱实例模型,如图 4 所示.

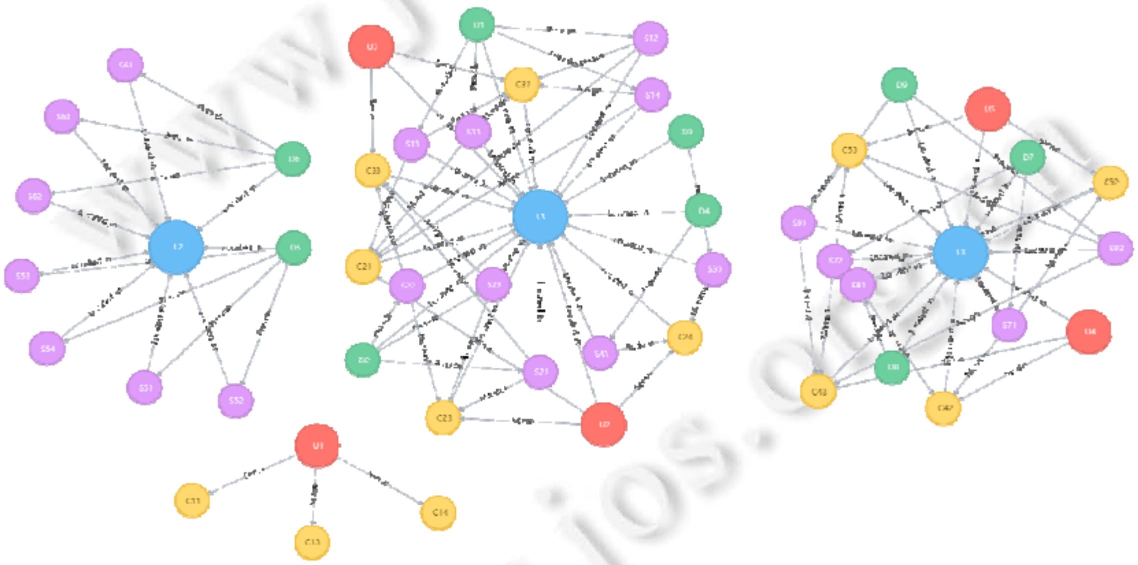

Fig.4 Instance model of the knowledge graph for smart home

图 4 智能家居知识图谱实例模型

5.2.2 情境感知服务的执行

为了维护智能家居知识图谱实例模型与场景实时信息的双向同步,实例模型自动持续更新状态.

(1) 更新概念实例,依次为位置实例、用户实例、设备实例、服务实例和环境实例.

(2) 更新关系实例,依次为"位于"实例、"监测"实例、"提高"实例、"降低"实例和"赋值"实例.

(3) 进行知识推理,依次为 Rule 1~Rule 4,重复执行步骤(1).

以 Jack 进入客厅时实例模型的状态变化为例,介绍智能家居情境感知服务的执行过程,如下所示.

- 首先,执行步骤 1.

更新 Jack(U_1)的位置信息,执行“$Get\ U_i.LName\rightarrow \boldsymbol{RTModel}(Get\ T_i.location)$”.此时,$U_1$:⟨$UName$:Jack,$LName$:sitting room⟩.更新 C_{11},C_{13} 和 C_{14} 的位置信息,执行“$Get\ C_i.LName\rightarrow Get\ U_j.LName$”(存在关系 $U_1\xrightarrow{Sense}C_{11}$, $U_1\xrightarrow{Sense}C_{13},U_1\xrightarrow{Sense}C_{14}$).此时,$C_{11}$:⟨$UName$:Jack,$LName$:sitting room,$CType$:Temperature,$CValue$:?,{$R_{\rm Min}$:19°C,$R_{\rm Max}$:26°C}⟩,$C_{13}$:⟨$UName$:Jack,$LName$:sitting room,$CType$:Brightness,$CValue$:?,{$R_{\rm Min}$:20%,$R_{\rm Max}$:*}⟩,$C_{14}$:⟨$UName$:Jack,$LName$:sitting room,$CType$:PM2.5,$CValue$:?,{$R_{\rm Min}$:*,$R_{\rm Max}$:35μg/m^3}⟩.

- 其次,执行步骤 2.

根据“$S_i.LName=C_j.LName\wedge S_i.CType=C_j.CType\wedge S_i.Effect=$“Monitor””,构造“监测”关系实例 $S_i\xrightarrow{Monitor}C_j$,得到 $S_{11}\xrightarrow{Monitor}C_{11},S_{21}\xrightarrow{Monitor}C_{13}$ 和 $S_{31}\xrightarrow{Monitor}C_{14}$;根据“$S_i.LName=C_j.LName\wedge S_i.CType=C_j.CType\wedge S_i.Effect=$“Increase””,构造“提高”关系实例 $S_i\xrightarrow{Increase}C_j$,得到 $S_{12}\xrightarrow{Increase}C_{11}$ 和 $S_{22}\xrightarrow{Increase}C_{13}$;根据“$S_i.LName=C_j.LName\wedge S_i.CType=C_j.CType\wedge S_i.Effect=$“Reduce””,构造“降低”关系实例 $S_i\xrightarrow{Reduce}C_j$,得到 $S_{13}\xrightarrow{Reduce}C_{11}$ 以及 $S_{41}\xrightarrow{Reduce}C_{14}$;根据“$S_i.LName=C_j.LName\wedge S_i.CType=C_j.CType\wedge S_i.Effect=$“Assign””,构造“赋值”关系实例 $S_i\xrightarrow{Assign}C_j$,得到 $S_{14}\xrightarrow{Assign}C_{11}$ 和 $S_{23}\xrightarrow{Assign}C_{13}$.

- 再次,执行步骤 3.

根据“$(\exists S_i)(\exists C_j)(S_i\xrightarrow{Monitor}C_j)\Rightarrow S_i.Status={\rm On}$”,执行“$Set\ S_{11}.status$=On$\rightarrow Set\ D_1.Key_1$=On”,“$Set\ S_{21}.status$=On$\rightarrow Set\ D_2.Key_1$=On”和“$Set\ S_{31}.status$=On$\rightarrow Set\ D_3.Key_1$=On”(存在关系 $D_1\xrightarrow{Provide}S_{11},D_2\xrightarrow{Provide}S_{21}$ 和 $D_3\xrightarrow{Provide}S_{31}$),即开启相应智能设备的监测功能.

- 然后,重复执行步骤 1.

更新 C_{11},C_{13} 和 C_{14} 的状态信息,执行“$Get\ C_i.CValue\rightarrow Get\ S_j.CValue\rightarrow Get\ D_k.Key_m$”(存在关系 $S_j\xrightarrow{Monitor}C_i\wedge D_k\xrightarrow{Provide}S_j$).此时,$C_{11}$:⟨$UName$:Jack,$LName$:sitting room,$CType$:Temperature,$CValue$:24°C,$R_{\rm Min}$:19°C,$R_{\rm Max}$:26°C⟩,$C_{13}$:⟨$UName$:Jack,$LName$:sitting room,$CType$:Brightness,$CValue$:30%,$R_{\rm Min}$:20%,$R_{\rm Max}$:*⟩,$C_{14}$:⟨$UName$:Jack,$LName$:sitting room,$CType$:PM2.5,$CValue$:40μg/m3,$R_{\rm Min}$:*,$R_{\rm Max}$:35μg/m^3⟩.

- 最后,执行步骤 3.

根据“$(\exists S_i)(\exists C_j)((S_i\xrightarrow{Reduce}C_j)\wedge(C_j.CValue>C_j.RMax))\Rightarrow S_i.Status={\rm On}$”,执行“$Set\ S_{41}.status$=On$\rightarrow Set\ D_4.Key_1$=On”(存在关系 $D_4\xrightarrow{Provide}S_{41}$),即开启空气净化器.

5.3 方法评估

为了验证方法的有效性,基于通用语言 Java 开发上述智能家居服务,并在服务开发过程、开发难度和执行性能等方面与本文方法进行比较.

5.3.1 智能家居情境感知服务的开发过程比较

基于通用语言 Java 进行服务开发,开发者必须熟悉不同智能设备的管理接口,并实现其交互.此外,由于服务建立在这些管理接口的基础上,其管理逻辑无法复用.

本文借助 SM@RT 工具[7,8]构造智能设备运行时模型,以统一的方式对设备进行数据读写和功能调用(SM@RT 源代码可以从文献[10]下载获得).SM@RT(supporting model at run time)是一种模型驱动的运行时软件体系结构构造方法及工具,包括特定领域建模语言(SM@RTlanguage)和代码生成器(SM@RT generator).SM@RT language 允许用户定义运行时软件体系结构的元模型及访问模型:元模型定义了目标系统的结构及可管理的元素;访问模型声明了元模型中管理这些元素的方法,即调用目标系统的 API.SM@RT generator 在元模型和访问模型的基础上,能够自动生成维护运行时软件体系结构的基础设施,将底层系统的实时状态反映到运行时模型.同时,智能设备运行时模型只需要构造 1 次,就能在不同的智能家居场景中复用.因此,基本不会带来

额外的工作量.

在智能设备运行时模型基础上,开发者仅声明面向场景的情境知识,配置概念实例与智能设备的映射关系,描述服务对象所处情境的约束条件,就能够自动构建智能家居情境感知服务.

5.3.2 智能家居情境感知服务的开发难度比较

表 10 对使用两种方法开发的智能家居情境感知服务的代码行数进行比较,其中,使用 Java 语言开发服务,每个服务独立开发,使用本文方法时,开发者则需要先实现面向场景的相关配置,即基础代码为 115 行.例如,C_{11} 是 Jack 的温度调节服务,Java 程序的代码行数为 220 行,其中,接口调用相关代码为 136 行,管理逻辑相关代码为 84 行,本文方法的新增配置行数为 6 行;C_{24} 是 Ken 的空气调节服务,Java 程序的代码行数为 140 行,其中,接口调用相关代码为 68 行,管理逻辑相关代码为 72 行,本文方法的新增配置行数为 6 行.Java 程序的平均代码行数为 186 行,而本文方法的平均配置行数为 16 行,其代码减少量超过 90%.

Table 10 Comparison in lines of code of two approaches

表 10 两种方法的服务代码行数比较

	Basic	C_{11}	C_{13}	C_{14}	C_{21}	C_{23}	C_{24}	C_{31}	C_{33}	C_{42}	C_{43}	C_{52}	C_{53}	Average
Java	0	220	198	140	220	198	140	220	197	163	188	163	188	186
Our approach	115	6	6	6	6	6	6	6	6	6	6	6	6	16

图 5 对使用两种方法的服务开发时间进行比较:使用 Java 语言开发单个服务的平均时间是 40min,且每个服务独立开发,开发时间随服务数量的增加而线性增长;使用本文方法时,开发者需要先花费固定时间配置面向场景的情境知识、概念实例与智能设备的映射关系,再根据服务需求,逐个配置服务对象所处情境的约束条件,配置每个约束的平均时间是 5min.因此,对于绝大多数智能家居场景,当智能设备充分发挥作用时(即服务达到一定数量),相对于使用通用语言进行服务开发,本文方法能够大幅度降低其开发时间.

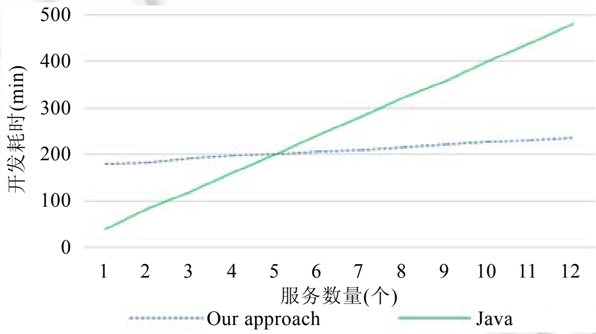

Fig.5 Comparison in development time of two approaches

图 5 两种方法的服务开发时间比较

5.3.3 智能家居情境感知服务的执行性能比较

为了比较两种方法的服务执行性能,在配置“CPU:3.1GHz;Memory:4GB”的系统环境下,执行不同数量的智能家居情境感知服务,并统计其平均响应时间,如图 6 所示.在 Java 程序中,每个服务顺序执行,分别执行管理逻辑并调用设备 API,因此,其平均响应时间随服务数量的增加而线性增长.使用本文方法时,知识图谱实例模型通过模型操作转换及设备 API 调用,实现与场景实时信息的双向同步,在此基础上,根据服务需求进行知识推理:一方面,由于需要额外操作来维护实例模型与智能家居场景的运行时同步,当服务数量少时,与 Java 程序相比,本文方法平均响应时间较高,从智能服务的角度看,这种性能上的差异是可以被接受的;另一方面,由于 Java 程序中服务独立执行,当服务达到一定数量时,会出现不同情境感知服务调用相同设备 API 获取场景信息的情况(例如,C_{11}、C_{21} 和 C_{31} 均要监测客厅温度),当存在大量服务时,与 Java 程序相比,本文方法能够有效降低设备 API 重复调用产生的额外开销,平均响应时间较低.

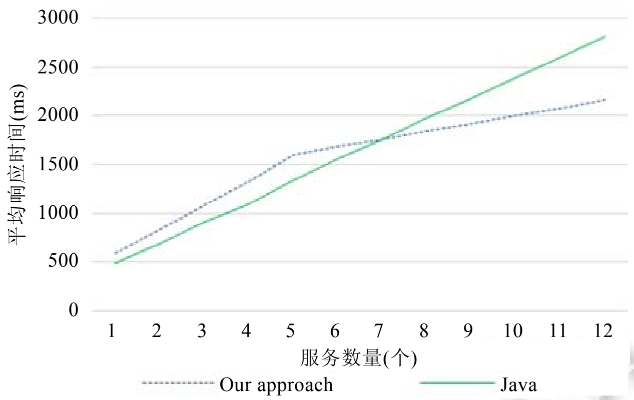

Fig.6 Comparison in execution time of two approaches

图 6 两种方法的服务执行性能比较

6 相关工作

在当前的物联网应用开发过程中,编程工作一般都接近操作系统级别,这要求程序员对底层系统的相关技术有非常深入的了解,使程序员将注意力集中在底层系统的相关问题上,而不是应用逻辑本身[11].存在一些研究工作,希望在保证效率的前提下,尽可能地简化物联网应用的开发过程.文献[12]提出一种面向物联网的认知管理框架,框架使用虚拟对象(virtual object)表示现实世界的对象,并使用组合虚拟对象(composite virtual object)表示一组可互操作的虚拟对象的聚合体,于是,可以面向组合虚拟对象开发物联网应用.文献[9,13]提出一种基于运行时软件体系结构模型的物联网应用开发方法,将设备能力抽象为运行时模型,通过模型转换实现应用场景模型与设备运行时模型的双向同步,于是,可以面向应用场景模型开发物联网应用.文献[14,15]提出一种基于 ECA 规则的物联网应用开发方法及支撑系统,给定一组面向应用场景的 ECA 规则,系统能够对其校验并自动生成面向底层系统的执行代码.文献[16]提出一种自顶向下的物联网应用开发方法及支撑系统,给定平台独立的应用管理逻辑,系统能够自动生成平台相关的具体配置和执行代码.此外,一些研究工作[17−19]提出了基于面向服务体系结构的解决方案,提供 RESTFul 服务形式的设备访问接口,以屏蔽底层系统的异构性和复杂性.尽管上述工作有效提高了物联网应用开发的抽象层次,但是开发者仍需面向场景手工编写物联网应用的管理逻辑.

为了支持物联网应用的服务组装和知识推理,一些研究工作提出将语义网的概念移植到物联网中,以提供物联网语义服务.文献[20]将注释嵌入到用 XML 表示的观测数据中,以准确地描述数据的语义,采用 OWL[21]建立物联网本体,并采用规则语言进行本体推论.文献[22]面向物联网领域提出统一的语义知识基础,包括资源、位置、环境、领域、规则和服务等本体,用于服务发现和组装.文献[23]面向物联网应用提出基于本体的语义建模与演化方法,包括通用上层本体以及一组用于领域本体扩展的柔性接口.上述工作在物联网传感数据、软件服务的基础上进行语义建模,但是缺乏其语义模型与底层数据、服务的联动机制.

北京大学团队在运行时模型理论及构造方法方面进行了研究[7,8,24−26]:给定系统元模型与一组管理接口,SM@RT 工具[10]就能自动生成代码,在保证性能的前提下实现模型到管理接口的映射;当系统元模型发生变化时,SM@RT 可以自动生成新的映射代码.我们进一步提出了基于运行时模型的物联网应用开发方法[9,13].在上述前期工作基础上,本文面向智能家居场景进行语义建模,并建立语义模型与底层系统的运行时同步机制.

7 总 结

智能家居情境感知服务需要面向场景进行开发,智能设备的多样性和智能服务的随需性,给其开发带来极大的难度和复杂度.本文提出一种智能家居情境感知服务的运行时建模与执行方法,通过知识图谱概念模型描述智能家居场景中概念和关系等抽象元素,通过运行时知识图谱实例模型表示智能家居情境知识,通过建立在

上述模型上的知识推理自动执行设备功能.于是,开发者仅声明面向场景的情境知识,配置概念实例与智能设备的映射关系,描述服务对象所处情境的约束条件,就能自动构建智能家居情境感知服务.本文方法能够极大地降低开发智能家居情境感知服务的难度和复杂度.

未来工作的重点主要包含以下两个方面:一方面,基于模型检测技术[27,28]对智能家居知识图谱实例模型进行深度分析,以保证实例模型和知识推理的正确性及完整性;另一方面,将方法运用到实际的智能家居场景中,并完善特定场景下的支撑机制.

References:

[1] Xu K, Wang XL, Wei W, Song HB, Mao B. Toward software defined smart home. IEEE Communications Magazine, 2016,54(5): 116−122. [doi: 10.1109/MCOM.2016.7470945]

[2] Atzori L, Iera A, Morabito G. The Internet of things: A survey. Computer Networks, 2010,54(15):2787−2805. [doi: 10.1016/ j.comnet.2010.05.010]

[3] Dixon C, Mahajan R, Agarwal S, Brush AJ, Lee B, Saroiu S, Bahl P. An operating system for the home. In: Proc. of the Usenix Conf. on Networked Systems Design and Implementation. Berkeley: USENIX Association, 2012. 25−25.

[4] Singhal A. Instroducing the knowledge graph: Things, not string, offical blog, of Google. 2012. http://goo.gl/zivFV

[5] Liu Q, Li Y, Duan H, Liu Y, Qin ZG. Knowledge graph construction techniques. Journal of Computer Research and Development, 2016,53(3):582−600 (in Chinese with English abstract). [doi: 10.7544/issn1000-1239.2016.20148228]

[6] Guan SP, Jin XL, Jia YT, Wang YZ, Cheng XQ. Knowledge graph oriented knowledge inference method: A survey. Ruan Jian Xue Bao/Journal of Software, 2018,29(10):2966−2994 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5551. htm [doi: 10.13328/j.cnki.jos.005551]

[7] Huang G, Song H, Mei H. SM@RT: Towards architecture-based runtime management of Internetware systems. In: Proc. of the Asia-Pacific Symp. on Internetware. ACM, 2009. 1−10. [doi: 10.1145/1640206.1640215]

[8] Song H, Huang G, Chauvel F, Xiong YF, Hu ZJ, Sun YC, Mei H. Supporting runtime software architecture: A bidirectional-transformation-based approach. Journal of Systems & Software, 2011,84(5):711−723. [doi: 10.1016/j.jss.2010.12.009]

[9] Chen X, Li AP, Zeng XE, Guo WZ, Huang G. Runtime model based approach to IoT application development. Frontiers of Computer Science, 2015,9(4):540−553. [doi: 10.1007/s11704-015-4362-0]

[10] Peking University. SM@RT: Supporting models at run-time. 2009. http://code.google.com/p/smatrt/

[11] Mottola L, Picco GP. Programming wireless sensor networks: Fundamental concepts and state of the art. ACM Computing Surveys, 2011,43(3):1−51. [doi: 10.1145/1922649.1922656]

[12] Vlacheas P, Giaffreda R, Stavroulaki V, Kelaidonis D, Foteinos V, Poulios G, Demestichas P, Somov A, Biswas RB, Moessner K. Enabling smart cities through a cognitive management framework for the internet of things. IEEE Communications Magazine, 2013, 51(6):102−111. [doi: 10.1109/MCOM.2013.6525602]

[13] Chen X, Zhang W, Huang G, Li AP, Guo WZ, Chen GL. Management approach of wireless sensor networks based on runtime model. Ruan Jian Xue Bao/Journal of Software, 2014,25(8):1696−1712 (in Chinese with English abstract). http://www.jos.org.cn/ 1000-9825/4665.htm [doi: 10.13328/j.cnki.jos.004665]

[14] Cano J, Rutten E, Benazzouz Y, Gurgen L. ECA rules for iot environment: A case study in safe design. In: Proc. of the IEEE 8th Int'l Conf. on Self-adaptive and Self-organizing Systems Workshops. Piscataway: IEEE, 2014. 116−121. [doi: 10.1109/SASOW. 2014.32]

[15] Chen YT, Chen CC, Chang HY, Lin HS, Chang HT. Constructing ECA rule for IoT application through a novel S2RG process: The exemplary ECA rules for smarter energy applications. In: Proc. of the Int'l Computer Symp. Piscataway: IEEE, 2017. 549−554. [doi: 10.1109/ICS.2016.0114]

[16] Guan GY, Dong W, Gao Y, Fu KB, Chen ZH. TinyLink: A holistic system for rapid development of IoT applications. In: Proc. of the Int'l Conf. on Mobile Computing and Networking. New York: ACM Press, 2017. 383−395. [doi: 10.1145/3117811. 3117825]

[17] Spiess P, Karnouskos S, Guinard D, Savio D, Baecker O, Souza LMSD, Trifa V. SOA-based integration of the Internet of things in enterprise services. In: Proc. of the IEEE Int'l Conf. on Web Services. Piscataway: IEEE, 2009. 968−975. [doi: 10.1109/ICWS. 2009.98]

[18] Janowicz K, Bröring A, Stasch C, Sachade S, Everding T, Llaves A. A RESTful proxy and data model for linked sensor data. Int'l Journal of Digital Earth, 2013,6(3):233−254. [doi: 10.1080/17538947.2011.614698]

[19] Google physical Web. 2018. http://google.github.io/physical-web/

[20] Sheth A, Henson C, Sahoo SS. Semantic sensor Web. IEEE Internet Computing, 2008,12(4):78−83. [doi: 10.1109/MIC.2008.87]

[21] Web ontology language (OWL). 2004. http://www.w3.org/TR/owl-ref/

[22] Nambi SNAU, Sarkar C, Prasad RV, Rahim A. A unified semantic knowledge base for IoT. In: Proc. of the Internet of Things. Piscataway: IEEE, 2014. 575−580. [doi: 10.1109/WF-IoT.2014.6803232]

[23] Ma M, Wang P, Chu CH. Ontology-based semantic modeling and evaluation for Internet of things applications. In: Proc. of the Internet of Things. Piscataway: IEEE, 2014. 24−30. [doi: 10.1109/iThings.2014.13]

[24] Huang G, Mei H, Yang FQ. Runtime recovery and manipulation of software architecture of component-based systems. Automated Software Engineering, 2006,13(2):257−281. [doi: 10.1007/s10515-006-7738-4]

[25] Mei H, Huang G, Lan L, Li JG. A software architecture centric self-adaptation approach for Internetware. Science China Information Sciences, 2008,51(6):722−742. [doi: 10.1007/s11432-008-0052-y]

[26] Song H, Xiong YF, Chauvel F, Huang G, Hu ZJ, Mei H. Generating synchronization engines between running systems and their model-based views. In: Proc. of the Int'l Conf. on MODELS in Software Engineering. Berlin: Springer-Verlag, 2009. 140−154.

[27] Zhang PC, Muccini H, Li BX. A classification and comparison of model checking software architecture techniques. Journal of Systems & Software, 2010,83(5):723−744. [doi: 10.1016/j.jss.2009.11.709]

[28] He X, Chen X, Cai S, Zhang Y, Huang G. Testing bidirectional model transformation using metamorphic testing. Information and Software Technology, 2018,104:109−129.

附中文参考文献:

[5] 刘峤,李杨,段宏,刘瑶,秦志光.知识图谱构建技术综述.计算机研究与发展,2016,53(3):582−600. [doi: 10.7544/issn1000-1239.2016.20148228]

[6] 官赛萍,靳小龙,贾岩涛,王元卓,程学旗.面向知识图谱的知识推理研究进展.软件学报,2018,29(10):2966−2994. http://www.jos.org.cn/1000-9825/5551.htm [doi: 10.13328/j.cnki.jos.005551]

[13] 陈星,张伟,黄罡,李隘鹏,郭文忠,陈国龙.基于运行时模型的无线传感网管理方法.软件学报,2014,25(8):1696−1712. http://www.jos.org.cn/1000-9825/4665.htm [doi: 10.13328/j.cnki.jos.004665]

陈星(1985－),男,福建永春人,博士,副教授,博士生导师,CCF 专业会员,主要研究领域为系统软件,软件自适应,云计算.

马郓(1989－),男,博士,CCF 专业会员,主要研究领域为系统软件,Web,移动计算,服务计算.

黄志明(1994－),男,硕士生,CCF 学生会员,主要研究领域为系统软件,软件自适应,软件中间件.

陈艺燕(1995－),女,硕士生,主要研究领域为自然语言处理.

叶心舒(1994－),女,硕士,主要研究领域为分布式系统,软件中间件,软件工程.

郭文忠(1979－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为计算智能及其在计算机网络中的应用研究.