大型模型克隆检测技术研究

梁正平[1,2] 谭佳加[1] 程一群[1] 马骁驰[1]
(深圳大学计算机与软件学院 深圳 518060)[1] (武汉大学软件工程国家重点实验室 武汉 430072)[2]

摘 要 模型克隆检测在软件维护、软件结构优化等方面具有重要价值和意义。首先综述了模型克隆的定义,接着对模型克隆的完整检测过程进行了详细划分和讨论,然后介绍了当前国际上最具代表性的两类大型模型克隆检测技术,最后对模型克隆检测的研究现状和亟需解决的问题进行了分析,并展望了该领域未来的研究方向。

关键词 模型驱动开发,模型克隆,克隆检测,子图同构,特征向量
中图法分类号 TP311.55 **文献标识码** A

Research on Clone Detection for Large-scale Model

LIANG Zheng-ping[1,2] TAN Jia-jia[1] CHENG Yi-qun[1] MA Xiao-chi[1]
(College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China)[1]
(State Key Laboratory of Software Engineering, Wuhan University, Wuhan 430072, China)[2]

Abstract Model clone detection is a very important technology for software maintenance, software structure optimization, etc. The paper first summarized the definition of model clone. Next, the overall detection process for model clone was divided and discussed in detail. After that, two kinds of typical clone detection technologies for large-scale model were introduced. Finally, the paper reviewed the state-of-art of current model clone detection research and identified some open research issues and pointed out possible future research directions in model clone detection area.

Keywords Model-driven development, Model clone, Clone detection, Sub-graph isomorphism, Characteristic vector

1 引言

软件开发实践中,由于大量"复制-粘贴-修改"操作的存在,以及受开发人员固有思维模式和开发能力的影响等,软件克隆现象普遍存在[1-4]。以代码克隆(Code Clone)为例,据相关文献统计,大型软件系统中约20%~50%的代码为克隆代码[5]。不论软件克隆的产生原因和负面影响如何,软件克隆存在的根源在于软件的不同部分之间存在内在的语义联系[6]。因此,对软件克隆进行检测和标识,发掘各克隆实例之间的语义关联,特别是发掘高层的结构克隆,有利于优化软件的结构和抽象层次,提高软件质量,同时降低软件维护的难度和成本。软件克隆检测可广泛应用于程序理解、软件重构、抄袭检测、版权保护、方面挖掘、产品线开发、恶意代码分析及错误预防等领域[1,4]。

作为新一代软件开发范式,模型驱动开发具有开发速度快、开发成本低、支持业务逻辑与技术细节的柔性桥接等众多优点[7]。随着体系结构、建模语言、CASE支持工具的日益成熟和标准化,模型驱动开发已广泛应用于现代各种大型软件系统的开发[8-10],特别是在嵌入式汽车软件等领域,以Matlab/Simulink为代码的建模语言和CASE工具已在相关软件产品的开发中占据了绝对的主流地位[11]。

模型是模型驱动开发中的核心软件制品,与传统软件开发中的代码类似,模型中同样存在着大量克隆[12]。由于模型比代码的抽象层次更高,因此对模型克隆(Model Clone)进行检测比传统的代码克隆检测在软件维护、软件结构优化等方面具有更高的价值和意义。相比代码克隆检测已取得的丰硕成果[13-19],模型克隆检测方兴未艾,是软件工程领域国际上当前最前沿的研究方向之一[20-23]。本文通过对模型克隆的定义、检测过程、现有的大型模型克隆检测技术等进行较全面的介绍,系统地分析和梳理模型克隆检测技术的研究现状,并对未来的发展方向进行讨论。

2 模型克隆的定义

近年来,有关软件克隆的研究十分活跃,ICSE[10]、ICSM[19]、IEEE Trans. on SE[5]、ACM Trans. on SE&M[17]等顶级学术会议和期刊都有大量相关研究进展和成果的报告。但对于什么是软件克隆,至今没有一个严格的形式定义[4]。较普遍接受的是早期Baxter等从相似性角度提出的代码克隆定义:符合某一相似性定义的相同或相似代码片段[24]。模型克隆的定义与代码克隆类似,通常模糊地定义为相同或相似的模型片段[23],但不同的研究者所给出的具体定义不同。

Deissenboeck等2008年在文献[10]中首次明确给出了模型克隆的定义。他们以Matlab/Simulink模型为例,通过提取Simulink模型中各基本模块的相关属性,将Simulink模型

到稿日期:2011-05-06 返修日期:2011-07-14 本文受国家自然科学基金项目(60903114),国家高技术研究发展计划(863)项目(2007AA01Z185),软件工程国家重点实验室开放基金项目(SKLSE20080702),深圳市科技研发资金基础研究计划项目(JC201005280432A)资助。
梁正平(1979—),男,博士,副教授,主要研究方向为模型克隆检测、需求建模与分析等,E-mail:liangzp@szu.edu.cn。

转换为纯粹数学意义上的有向标记图 G。称 G 中两个子图 G_1 和 G_2 所对应的模型片段 f_1 和 f_2 为克隆对(Model Clone Pair),当且仅当(1)G_1 和 G_2 均为连通图,(2)G_1 和 G_2 没有重叠顶点,(3)G_1 和 G_2 就其顶点和边的标记具有同构关系。在此基础上,Deissenboeck 等提出了克隆类(Clone Class)的定义,称两两具有克隆对关系的所有模型片段构成一个克隆类。值得特别说明的是,Deissenboeck 等同时认为,某些仅有一两条边不具有对应关系的模型片段可能也与克隆相关。为此,他们在实际检测过程中将克隆类的定义弱化为只需顶点同构,而不需边同构。

基于同构的定义非常严格,模型驱动开发中很多事实上存在克隆关系的模型片段可能并不满足 Deissenboeck 等提出的定义。为支持模型克隆检测结果召回率(Recall)的提高,Pham 等 2009 年在文献[21]中将模型克隆的定义划分为精确克隆(Exact Clone)和相似克隆(Similar Clone)两类。Pham 等的定义借鉴了 Deissenboeck 等将模型先转换为有向标记图的方式,其克隆组(Clone Group)的定义也类似于Deissenboeck 等的克隆类,即两两具有克隆对关系的所有模型片段构成一个克隆组。Pham 等对精确克隆对的定义与 Deissenboeck 的克隆对定义基本相同,只是没有强调 G_1 和 G_2 必须为连通图。相似克隆对的定义则基于文献[25]所提出的 Exas 特征向量,具体如下:设分别有 k 和 h 条边的模型片段 f_k^1 和 f_h^2 所对应子图 G_1 和 G_2 的 Exas 向量分别为 v_1 和 v_2,称模型片段 f_1 和 f_2 为相似克隆对,当且仅当(1)G_1 和 G_2 没有重叠顶点,(2)向量 v_1 和 v_2 的距离不超过某一阈值 δ,(3)G_1 和 G_2 中存在一个边数为 s 的精确克隆子图对 G_1^s 和 G_2^s,且 s/k 和 s/h 均不小于某一阈值 σ。

上述模型克隆定义以图模型的结构相似为视角。Gold 等 2010 年在文献[22]中认为,对于一些建模语言,如 Max/MSP 等,由于模型的语义还与模型中元素的空间布局相关,因此两个模型片段是否构成克隆关系,除需考虑它们的结构是否相似外,还需考虑模型中元素的空间布局在多大程度上相似。Gold 等虽没给出具体的模型克隆定义,但他们的观点为模型克隆的研究提供了更为宽广的视野,即两个模型片段是否构成克隆关系,不仅与其语法相关,也与其语义相关。

3 模型克隆的检测过程

全面的模型克隆检测必须能找出所有符合克隆定义的模型片段,并对它们进行有效的分类、组织和展示。由于事先不知道哪些模型片段可能与克隆相关,因此在克隆检测过程中必须两两比对模型中所有的原始片段。对于大型模型而言,这种方式显然在计算复杂度上难以接受。为此,在执行实际的克隆比对之前,通常先对模型进行某种形式的预处理和规范化,以减少各种外围噪音对克隆检测的干扰,进而减少所需的比对次数并降低比对难度。此外,为提高检测结果的精确率(Precision)和可用性,还需对检测出的模型克隆对进行后期的分析、过滤和格式化等。完整意义上的模型克隆检测过程包括 8 个前后关联的阶段,如图 1 所示。

预处理是模型克隆检测的初始阶段,其处理对象为待检测的原始模型。预处理通常采用如下 3 种形式对原始模型进行合理简化、组合或分解:①删除模型图中与克隆检测无关的内容,如各类注释信息,减少模型元素的数量;②将具有层次关系的模型图进行扁平化处理,减少模型元素的数量,并为后阶段候选克隆片段的选取提供便利;③提取模型图中输入/输出数大于给定阈值的块元素,或对模型图中的层次关系进行分解,将复杂模型转换为多个简单模型。

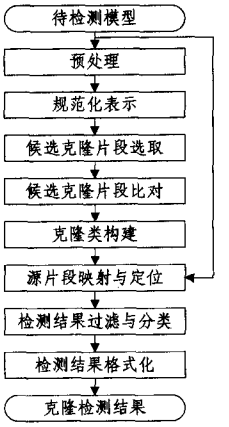

图 1 模型克隆检测过程

规范化表示阶段将预处理后的模型转换为某种较单一、标准的中间形式。规范化表示最常见的形式是将模型转换为带属性的有向标记图。图中的顶点对应模型的基本模块,图中的有向边对应模块间的数据流或控制流,同时抽取模型中模块的属性作为图中顶点的标记。规范化表示需关注的核心问题是规范化的程度,若过于规范化,容易导致检测结果中出现大量的假阳性,从而降低克隆检测的精确率;若规范化程度不够,则可能导致很多克隆类,特别是各种相似克隆类无法检测到,从而降低克隆检测的能力和召回率。

选取候选克隆片段是进行克隆检测的前提条件,也是决定整体克隆检测速度的关键因素之一。对于具有 n 条边的模型,若采用穷举方式,理论上最多存在 2^n-1 个候选克隆,仅检测它们两两之间的克隆对关系,即需进行 $C_{2^n-1}^2$ 次比对计算,若再进一步构建克隆类,则计算量更为庞大。为此,在候选克隆片段的选取阶段通常引入启发式策略,从源头上减少候选克隆片段的数量,以缩短克隆检测的时间和降低空间复杂度。

比对候选克隆片段是模型克隆检测的核心内容之一。若两个模型片段符合克隆的定义,则称它们构成一个克隆对。主要的克隆比对形式可分为两大类型:①基于模型片段所对应子图的拓扑关系;②基于模型片段所对应特征向量的距离。前者能用于判断模型片段间的精确克隆关系,后者则还可用于相似克隆的判断。克隆片段比对的速度也是决定整体克隆检测速度的关键因素。

构建克隆类是模型克隆检测的另一核心内容。仅检测出所有的克隆对尚不足以展现出模型整体的克隆全貌,还需将相关的克隆对进行聚类,以构建各种模型克隆类。极大团覆盖算法[24]是较常采用的克隆类聚类算法。克隆类构建过程中相对较难处理的问题是不同克隆类之间存在重叠的情形,特别是元素粒度不同的重叠克隆类还存在元素间包含关系的情形。

源片段映射与定位阶段的输入既包括在模型规范化表示形式上检测出的克隆对和克隆类,也包括原始待检测模型。本阶段首先构建模型的规范化表示到原始模型的映射,再基于上一阶段的检测结果在原始模型中定位相应的克隆对和克隆类。本阶段与模型规范化表示阶段形成互补。

检测结果过滤与分类阶段的目标是提高检测结果的精确

率。由于预处理和规范化表示阶段对待检测模型进行了抽象，在此基础上获得的检测结果中可能存在某些假阳性和不相关性，因此需对它们进行合理的过滤与分类。本阶段的主要任务是设计有效的过滤与分类规则，并将其与克隆检测的动机和目标关联，以提高检测结果的精确率。

模型克隆检测最后阶段的任务是用直观、易于接受和理解的形式对检测结果进行格式化展现。较好的格式化形式包括在原始模型中采用不同颜色对存在克隆关系的片段进行着色、采用 HTML 语言构建模型克隆片段间的链接关系、采用 XML 形式对存在克隆关系的模型片段进行描述等。

值得说明的是，上述 8 个阶段是对模型克隆检测过程一般意义上的线性细分，实际的克隆检测过程中，通常并非严格如此。可能前后关联的两个甚至多个阶段糅合在一起；如将候选克隆片段的选取和比对糅合在一起，也可能忽略了某个阶段的内容，如没有对检测结果进行过滤和分类等。

4 基于子图同构的模型克隆检测

基于子图同构的模型克隆检测面向精确克隆的检测，最具代表性的是 Deissenboeck 等[10]的 CloneDetective 和 Pham 等[21]的 eScan。CloneDetective 和 eScan 对精确克隆的定义基本相同，且均以 Matlab/Simulink 模型为验证对象。

4.1 CloneDetective

CloneDetective 将模型克隆检测分为预处理和规范化、克隆对检测、克隆聚类、检测结果可视化等 4 个阶段。预处理和规范化阶段首先读取待检测模型，对其进行扁平化处理，同时删除模型中未连接的线，然后将模型转换为一个有向标记图 $G=(V,E,L)$，其中顶点集 V 对应模型中的块集，有向边集 $E\subset V\times V$ 对应模型中的数据线，标记函数 $L:V\cup E\rightarrow N$ 将 G 中的顶点集和边集映射到一个规划化的标记集 N，N 为待检测模型中块和线的属性所构成的集合。

克隆对检测是 CloneDetective 方法的核心，其最大的特色是引入了一个轻量级的启发式策略，以降低克隆检测的时间复杂度。具体而言，CloneDetective 采用深度优先的方式，首先选取一对标记相同，且尚未被检测的顶点，然后沿图中的有向边逐一对它们进行扩展，且要求每次通过扩展操作新加入的顶点对和边对均具有相同的标记，直到所生成的子图对无法继续扩展为止，此时所生成的子图对即为克隆对。随后重复这一过程，直至检测出全部克隆对。此过程中的关键技术是引入了一个启发式的相似性函数(Similarly Function)σ：$V\times V\rightarrow[0,1]$，该函数以递归形式，根据待扩展顶点对及各自后续顶点的标记，给每一待扩展顶点对赋予一个相似度值。相似度值越高的待扩展顶点对表示该顶点对与克隆相关的可能性越大。实际检测过程中，对于可从某顶点对扩展出的多对顶点，仅选取其中相似度值最高的顶点对进行扩展，其它顶点对则直接剪除。该方式可将克隆检测的时间复杂度从指数级降低至多项式级。

CloneDetective 在克隆聚类阶段的主要工作是对两个细节问题进行了专门处理：一是将克隆类中子图的严格同构弱化为只需顶点同构，而不需边同构，以将部分仅有一两条边不具对应关系，但整体可能与克隆密切相关的子图也纳入相关克隆类中；二是通过检查克隆对之间的包含关系来构建被粗粒度元素克隆类所隐藏的细粒度元素克隆类。为了检测结果的可视化，通过创建一个可为模型的块元素着色的 Matlab 脚本，在 Matlab/Simulink 环境中实现了对 Matlab/Simulink 模型中存在的克隆进行可视化标注和区分的方法。

采用启发式扩展策略，可使 CloneDetective 在时间方面具有较理想的效果，其成为世界上首个能支持大型模型克隆检测的方法。但该启发式策略将大多数分支在子图扩展过程中进行了剪除，导致 CloneDetective 检测结果的召回率较低。同时，CloneDetective 仅支持极大克隆模型片段的检测，对于超大型模型，无法灵活限定克隆检测的时间。

4.2 eScan

eScan 的模型克隆检测分为预处理和规范化、克隆对检测、克隆分组、结果过滤等 4 个阶段。eScan 没有专门考虑检测结果的报告方式，且其预处理和规范化直接借用 CloneDetective 的技术。

eScan 进行克隆检测的出发点是边数为 k 的模型克隆片段对必然可由边数为 $k-1$ 的模型克隆片段对通过扩充某对边得到。对此，eScan 采用穷举方式列出对应各模型片段的所有子图，并引入格的形式组织所有子图，称之为克隆格(Clone Lattice)。其中，格中的顶点表示子图，格中的有向边表示子图间相差一条边的扩展关系，且边数相同的子图所对应顶点在格中处于同一层。

与 CloneDetective 类似，克隆对检测也是 eScan 方法的核心。eScan 从克隆格的最底层开始，先检测该层顶点所代表子图间的克隆关系，并对该层中具有克隆关系的顶点进行初始分类。在此基础上，采用深度优先的方式，依次从最底层的各初始克隆类出发，逐一建立克隆格中相邻层相关顶点的扩展关系，并在每个扩展环节基于克隆定义对新扩展的顶点进行初始分类处理。此过程中的关键技术包括两个方面：一是采用规范化标记(Canonical Labeling)[26]对子图进行唯一标注，并据此判断两个子图是否同构；二是基于规范化标记为每个子图定义一个唯一的双亲子图，避免检测过程中对某些顶点进行重复扩展，减少无谓的时间开销。

克隆分组阶段 eScan 逐层划分克隆格中与克隆相关的顶点，构建各种克隆类。划分的依据依然是各顶点所对应的规范化标记，即将具有相同规范化标记的顶点归入同一个克隆类。为处理克隆格中具有相同规范化标记的两个顶点所对应的子图因存在重叠而并不构成克隆关系的问题，eScan 在此引入了极大团覆盖算法 Bron-Kerbosch[27]。对于结果的过滤，eScan 仅考虑了若一个克隆类被另一个克隆类覆盖，则在结果中将被覆盖的克隆类去除。

相比 CloneDetective，eScan 因采用穷举的方式而具有更好的召回率。引入唯一双亲子图技术，以使 eScan 亦能较好地支持大型模型的克隆检测。此外，eScan 采用格的形式对各模型片段进行分层组织，可实现递增式的克隆检测，支持克隆检测时间的灵活设定。

5 基于特征向量的模型克隆检测

基于特征向量的模型克隆检测通过提取待检测模型片段的特征集，构建相应的特征向量。然后以特征向量间的距离作为判断对应模型片段间是否具有克隆关系的依据，既能用于精确克隆的检测，也能用于相似克隆的检测，最具代表性的是 Pham 等的 aScan[21]。

aScan 的检测过程分为 4 个阶段，与 eScan 完全一样，且同样采用克隆格的形式对模型片段进行组织。两者的不同之

处在于 aScan 的后 3 个阶段融合在一个大的迭代型循环中，而 eScan 的后 3 个阶段总体上具有先后顺序，同时两者在克隆对检测、克隆分组阶段的操作细节上存在较大差异。

在克隆对检测之前，aScan 首先提取模型片段的各种(p, q)结点和 n 路径，并基于这些结构模式的出现次数构建模型片段的 Exas 特征向量。Exas 特征向量具有非常好的性质，能确保两个特征向量的距离小于它们所对应模型片段编辑距离的常数倍。在此基础上，基于给定的阈值，采用广度优先的方式从克隆格的最底层开始逐一检测当前层和前面若干层中各顶点的克隆关系，建立与克隆格当前层对应的初始克隆类，再扩展当前层中与克隆相关的片段，直至完成所有模型片段的检测。aScan 在此过程中采用的关键技术包括两个方面：一是引入候选窗口的概念来限定克隆格中当前层的元素需与前面哪些层的元素进行比对；二是引入启发式的剪枝技术，以进一步减少候选模型克隆片段的数量，降低克隆检测的时间复杂度。

对于克隆分组，aScan 最大的特色是引入位置敏感哈希函数(Locality Sensitive Hashing)[28]对各初始克隆类进行划分，然后在划分的基础上采用极大团覆盖算法 Bron-Kerbosch 构建克隆类。位置敏感哈希函数一方面支持高维数据的快速分类，另一方面能以事先设定的概率保证散列前的高维数据经过哈希之后，能够在一定程度上相似，因而特别适合用于表示模型特征的高维向量比对与划分。由于在迭代过程中及时过滤了克隆类，因此 aScan 的过滤操作比 eScan 更简单，只需去除前一层中被当前层覆盖的克隆类即可，而不需考虑属于更前层次的克隆类。

aScan 采用与 eScan 相同的递增式克隆检测，同样支持克隆检测时间的灵活设定。相比 CloneDetective 和 eScan，aScan 最大的优点是支持相似克隆的检测，因而检测能力更强。

6 模型克隆检测技术讨论

相比软件克隆研究领域已取得丰硕成果的代码克隆研究，模型克隆的研究方兴未艾。综观软件克隆检测领域的整体研究现状和发展动态，不难发现：

首先，模型克隆检测已成为软件克隆检测领域的研究热点和前沿，但尚处于萌芽状态，技术上还存在许多急需解决和改进的地方。如模型克隆检测速度的提高、克隆检测可扩展性的提高、克隆检测结果精确率和召回率的提高、模型相似克隆检测技术的改进、模型结构克隆的检测方法、模型克隆检测技术的对比研究、大型模型克隆检测的经验研究和实例分析等。

其次，现有的各类代码克隆检测技术为模型克隆检测提供了众多可借鉴的研究方法和途径。例如基于形符技术中的归一化思想、基于抽象语法树和基于程序依赖图技术中的子树/子图快速检测算法、基于度量技术中的特征抽取和近似克隆检测方法等。但由于模型一般为二维，甚至三维的图结构，与线性化的一维代码序列存在较大差异，代码克隆检测中已成功运用的各项技术一般难以直接应用于模型的克隆检测。

再次，基于图的大型模型克隆检测必须突破传统的子图同构思路。子图同构问题属 NP 难题[29]，基于子图同构的模型克隆检测技术在检测速度和可扩展性上存在难以突破的瓶颈。同时，传统的子图同构思路难以检测各种类型的语法相似模型克隆和语义等价模型克隆。

另外，现有的模型克隆检测技术主要面向以 Matlab/Simulink 为代表的数据流模型，尚不能直接用于各种类型的状态模型和过程模型，如 State Charts、BPEL 等。

因此，模型克隆检测研究必须探索新的思路和方法，并充分发掘待检测模型的内在特性，在降低模型克隆检测复杂度的同时提高模型克隆检测的综合能力。近期可能取得突破的研究途径包括改进模型的预处理和规范化表示形式以设计通用型模型克隆检测技术、采用分布式技术提高模型克隆检测的可扩展性、采用特征指纹技术提高克隆检测的速度、采用数据挖掘技术提高克隆检测的召回率、采用目标导向的清洗机制提高克隆检测的精确率、采用人工注入克隆方式改进克隆检测结果的分析能力等。

结束语 模型克隆检测是一种重要的软件分析技术，对于软件维护、软件结构优化等具有重要的价值和意义，有利于提高软件产品的质量，降低软件开发的成本。作为当前软件工程领域最前沿的研究课题之一，模型克隆检测的研究已获得国际学术界和产业界众多研究人员和机构的极大关注。

本文从模型克隆的定义、模型克隆的检测过程出发，讨论了模型克隆检测所需解决的主要问题。通过对现有大型模型克隆检测技术的介绍和分析，对模型克隆检测的研究现状进行了综述，在此基础上对模型克隆检测领域的发展趋势进行了分析和讨论。

参 考 文 献

[1] Roy C K, Cordy J R. A Survey on Software Clone Detection Research[R]. 2007-541. Queen's University at Kingston, 2007

[2] Bellon S, Koschke R, Antoniol G, et al. Comparison and Evaluation of Clone Detection Tools[J]. IEEE Trans. on Software Engineering, 2007, 33(9): 577-591

[3] Roy C K, Cordy J R, Koschke R. Comparison and Evaluation of Clone Detection Techniques and Tools: A Qualitative Approach [J]. Science of Computer Programming, 2009, 74(7): 470-495

[4] Pate J R, Tairas R, Kraft N A. Clone Evolution: a Systematic Review[R]. SERG-2010-01. The University of Alabama, 2010

[5] Basit H, Jarzabek S. A Data Mining Approach for Detecting Higher-level Clones in Software[J]. IEEE Trans. on Software Engineering, 2009, 35(94): 497-513

[6] Jarzabek S, Xue Y. Are Clones Harmful for Maintenance? [C]// Proc. of the 4th Int'l Workshop on Software Clones. 2010: 73-74

[7] Schmidt D C. Guest Editor's Introduction: Model-driven Engineering[J]. IEEE Computer, 2006, 39(2): 25-31

[8] 蒋哲远，蒋建国. 面向服务领域软件系统的模型驱动建模方法[J]. 计算机科学，2008，35(5)：274-279

[9] 任磊，王威信，等. 种模型驱动的交互式信息可视化开发方法[J]. 软件学报，2008，19(8)：1947-1964

[10] Deissenboeck F, Hummel B, Juergens E, et al. Clone Detection in Automotive Model-based Development[C]// Proc. of the 30th Int'l Conf. on Software Engineering. 2008: 603-612

[11] Karris S. Introduction to Simulink with Engineering Applications(Second Edition)[M]. Orchard Publications, 2008

[12] Deissenboeck F, Hummel B, Juergens E, et al. Model Clone Detection in Practice[C]// Proc. of the 4th Int'l Workshop on Software Clones. 2010: 57-64

[13] Kamiya T, Kusumoto S, Inoue K. CCFinder: A Multilinguistic Token-based Code Clone Detection System for Large Scale Source Code[J]. IEEE Trans. on Software Engineering, 2002, 28(7): 654-670

(下转第 52 页)

4,5 和 7 上,也部署了邻居发现应用任务,可以和其它应用任务结合感知融合数据,然后一起发送到基站或网关。在节点 1,2,5 和 8 上,部署动态路由应用任务,记录感知数据包在网络中传递的路由路径,采用了消息的捎带技术。可以看出节点 1 和 2 的父节点都是节点 4,而节点 7 又是节点 4 和 5 的父节点,最后都通过节点 9 到达基站节点 0,而节点 8 是通过父节点 10 到达基站节点 0。因此,可看出整个网络由两棵树组成。

表 1 应用任务的感知数据

节点号	热敏电阻值	光敏电阻值	麦克风值	邻居节点集	动态路由路径
1	0x01C6	未部署	未部署	未部署	1→4→7→9→0
2	0x01C6	未部署	未部署	未部署	2→4→7→9→0
3	未部署	0x037D	未部署	未部署	未部署
4	0x01C5	未部署	未部署	(6,1,5,2,7)	未部署
5	未部署	0x0369	未部署	(4,3,2,8,7)	5→7→9→0
6	0x01C4	未部署	未部署	未部署	未部署
7	未部署	未部署	未部署	(4,5,9,10)	未部署
8	未部署	0x037A	未部署	未部署	8→10→0
9	未部署	未部署	0x01D8	未部署	未部署
10	未部署	0x0373	0x01CA	未部署	未部署

验证的测试结果表明,各种应用任务都可以并行运行,并且可以根据用户的消息随时调整。开发人员也可以根据 WSNs 中间件提供的接口开发出多种不同的应用任务。在整个的验证测试过程中,WSNs 中间件运行稳定。

结束语 本文介绍了 WSNs 的发展趋势,系统更加复杂,承载的应用也更加多样,简单的 WSNs 中间件很难适用于未来多种 WSNs 应用的需求。提出了一种支持多应用任务的 WSNs 中间件,并详细地描述了中间件的整体设计、组件划分和实现过程。通过实物节点的组网模拟实验,成功地验证了整个中间件系统和所承载应用任务能够稳定地工作。本中间件的设计与实现为进一步研究 WSNs 理论与技术奠定了平台基础。

参 考 文 献

[1] Akyildiz I F, Su W, Sankarasubramaniam Y, et al. Wireless sensor networks: a survey[J]. Computer Networks, 2002, 38(4): 393-422

[2] Wang M M, Cal J N, Li J, et al. Middleware for wireless sensor network: A survey[J]. Journal of Computer Science and Technology, 2008, 23(3): 305-326

[3] Levis P, Culler D. A tiny virtual machine for sensor networks [C]// Proc of the 10th Int'l Conf on Architectural Support for Programming Languages and Operating Systems (ASPLOSX). New York: ACM Press, 2002: 85-95

[4] Madden S R, Franklin M J, Hellerstein J M. TinyDB: An acquisitional query processing system for sensor networks[J]. ACM Trans on Database Systems, 2005, 30(1): 122-173

[5] Fok C, Roman G, Lu C. Mobile agent middleware for sensor networks: An application case study[C]// Proc. the 4th Int. Conf. Information Processing in Sensor Networks (IPSN). Los Angeles, California, USA, Apr. 2005: 382-387

[6] Heinzelman W B, Murphy A L, Carvalho H S, et al. Middleware to support sensor network application [J]. IEEE Networks, 2004, 18(1): 6-14

[7] Crossbow MICAz Mote Specification [EB/OL]. http://www.cro-ssbow.com

[8] Crossbow sensor platform[EB/OL]. http://www.crossbow.com

(上接第 31 页)

[14] Li Z, Lu S, Myagmar S, et al. CP-Miner: Finding Copy-Paste and Related Bugs in Large-scale Software Code[J]. IEEE Trans. on Software Engineering, 2006, 32(3): 176-192

[15] Falke R, Frenzel P, Koschke R. Empirical Evaluation of Clone Detection using Syntax Suffix Trees[J]. Empirical Software Engineering, 2008, 13(6): 601-643

[16] Evans W S, Fraser C W, Ma F. Clone Detection via Structure Abstraction[J]. Software Quality Journal, 2009, 17(4): 309-330

[17] Duala-Ekoko E, Robillard M. Clone Region Descriptors: Representing and Tracking Duplication in Source Code [J]. ACM Trans. on Software Engineering and Methodology, 2010, 20(1): 1-31

[18] Roy C K, Cordy J R. Near-Miss Function Clones in Open Source Software: An Empirical Study[J]. Journal of Software Maintenance and Evolution: Research and Practice, 2010, 22(3): 165-189

[19] Hummel B, Juergens E, Heinemann L, et al. Index-based Code Clone Detection: Incremental, Distributed, Scalable[C]// Proc. of the 26th IEEE Int'l Conf. on Software Maintenance. 2010: 1-9

[20] Liu H, Ma Z, Zhang L, et al. Detecting Duplications in Sequence Diagrams Based on Suffix Trees[C]// Proc. of the 13th Asia Pacific Software Engineering Conf.. 2006: 269-276

[21] Pham N H, Nguyen H A, Nguyen T T, et al. Complete and Accurate Clone Detection in Graph-based Models[C]// Proc. of the 31th Int'l Conf. on Software Engineering. 2009: 276-286

[22] Gold N, Krinke J, Harman M, et al. Issues in Clone Classification for Dataflow Languages[C]// Proc. of the 4th Int'l Workshop on Software Clones. 2010: 83-84

[23] Störrle H. Towards Clone Detection in UML Domain Models [C]// Proc. of the 4th Euro. Conf. on Software Architecture. 2010: 285-293

[24] Baxter I, Yahin A, Moura L, et al. Clone Detection Using Abstract Syntax Trees[C]// Proc. of the 14th IEEE Int'l Conf. on Software Maintenance. 1998: 368-377

[25] Nguyen H A, Nguyen T T, Pham N H, et al. Accurate and Efficient Structural Characteristic Feature Extraction for Clone Detection[C]// Proc. of the 12th Int'l Conf. on Fundamental Approaches to Software Engineering. 2009: 440-455

[26] Kuramochi M, Karypis G. Finding Frequent Patterns in a Large Sparse Graph[J]. Data Mining and Knowledge Discovery, 2005, 11(3): 243-271

[27] Bron C, Kerbosch J. Algorithm 457: Finding All Cliques of an Undirected Graph[J]. Communications of the ACM, 1973, 16 (9): 575-577

[28] Andoni A, Indyk P. Near-optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions[J]. Communications of the ACM, 2008, 51(1): 117-122

[29] Zou Z, Li J, Gao H, et al. Mining Frequent Subgraph Patterns from Uncertain Graph Data[J]. IEEE Trans. on Knowledge and Data Engineering, 2010, 22(9): 1203-1218