

九十年代初CASE 的研究现状

R.J. Norman, W. Stevens 等

1990年12月,在美国加州Irvine召开了第四届CASE国际专题讨论会。会上交流总结了当前计算机辅助软件工程(Computer Aided Software Engineering, CASE)的研究现状。由Norman等人成文,并在1991年5月美国德州Austin召开的第十三届ICSE上,向大会汇报了CASE状况的450个论点。本文即据此视国内情况编译而成。下届CASE国际专题讨论会将于1992年7月在加拿大魁北克省Montreal召开。

一、现有系统

1.1 维护与管理

正在进行的:

- 下列的成熟工具:更改控制、动态分析、测试、静态分析、设计恢复、公文包分析/系统清单、配置管理、构造工具(代码重构)、过程管理、数据有理化、错误跟踪、自动建档;

- 弄清工具后面的方法,工具实现滞后;
- 上述下划线工具未达到期望的所有目的。

尚未进行的:

- 集成式维护支持环境;
- 维护活动管理工具;
- 维护活动方法研究;
- 维护管理。

需要的:

- 下述工具:设计恢复(改进与提高)、规格说明的恢复,正向工程期间设计历史的记录、后继维护、测试、与库的集成;
- 更有效的专用测度;
- 现有测度的更多使用;

- 上面下划线项的更好工具。

应该调查研究的:

- 维护活动的过程模型;
- 更多的方法与工具,用以记录设计过程中所考虑的假设和(已放弃的)选择;
- 维护工具集成的方法。

总的看法

- 维护被看成耗费而不是投入;
- 维护不单是代码的更改,而应是系统行为的更改;
- 维护是前进,不是倒退;
- 已有一些有前途的研究与开发原型的维护工具箱;
- 有两种不同类型的维护:错误更正与系统性能增强。

1.2 逆向工程与设计恢复

正在进行的:

- 对建立基本静态表示的支持,如调用/被调用、部件依赖性;

需要的:

- 各种分析技术,用以支持基于以下两种类型的逆向工程:一是基于系统的种类(如嵌入式实时系统,并发与顺序体系结构系统、管理信息系统等),二是基于可用信息的类型(如非存在的设计文档、与当前实现不一致的设计描述等);
- “构作时”的表示法与记法,用以依待恢复过程的抽象设计级别,记录动态和静态描述,它们与正向工程的记法不同;
- 需要记录部分的或不完整的信息和与待恢复对象有关的恢复过程的状态;
- 需要建立正、反向工程与表示之间的联系;

- 详细图形记法的群集(例如被恢复系统的平面数据流程图);
- 广泛的记法,从详细的到抽象的;
- 用“冲突调停”支持自底向上和自顶向下的分析;
- 动态分析应具有发现不可达代码和死代码的能力;
- 必须认识到在一现存系统的恢复过程中,基本系统是渐进而非静态的;需要不断引入恢复的信息,且需要时修改之;
- 需要逆向工程支持小组维护或确认系统以及了解应用,也需要支持小组从一现存系统创建新的系统;
- 恢复过程产生多种数据类型,有显式的和导出的;保留机制的复杂性随系统规模增大而增加;必须从存储和性能着眼,考虑效率
- 逆向工程应与测试计划、质量/可靠性模型、安全性分析、评价测度等集成在一起;
- 应能恢复系统实现中记录的行业规则,以便重新建立的规格说明能表示系统存在和支持该组织时的真实性;
- 从商业观点出发,需要支持判断一系统是否要进行逆向工程,将出现什么级别的逆向工程,以及将会是什么类型的逆向工程。为此要求回答如下问题:成本?当前体系结构的功能评价?数据冗余/集成的级别?信息寿命?给定系统评价和支持该过程的可用工具,逆向工程所需的时间(把握市场时机)?在执行逆向工程时,是否可获得当前可用的信息?
- 需要工具来保证恢复的设计是正确的;
- 逆向工程应视为生存期中“计划废弃”的一部分。

应调查研究的:

- 对于有多版本存在的系统,研究实现其设计恢复的方法。

1.3 利旧软件重用

通过讨论,确定了“利旧(salvaging)重用”的定义,是指以某种类似或新的方式,在新系统的开发中重用现存的代码或设计。

“CASE”的定义指将软件工程技术 和 计算机技术商业性地广泛应用于软件的开发过程。

正在进行的:

- 当充分了解领域时,领域分析对重用的帮助作用;
- 代码库系统;
- 重构程序的工具;
- 通用维护工具;
- 在某些情况下,可以最低限度抽象的增量利旧方式完成重用;
- 在领域专家用得上的情况下,由底向上和由顶向下的领域分析;
- 主要在商业领域的有限应用生成;
- 体系结构及大型系统部件级的重用计划;
- 经过设计抽象,体系结构的端口;
- 以记录表格、拟合、功能等方式记录的设计重用;
- 现存代码段的小规模重用;
- 系统部件的重用/重工作;
- 通过设计抽象库,实现人工或自动(事实陈述)重用,软件重用问题类似于CASE环境的构造;
- 经过计划、实现和适当完成后,事后调查研究允许有限的设计决策文档帮助今后的维护和系统部件重用;
- 事后调查研究是获取设计决策的有效手段。

尚未进行的:

- 领域分析过程中更高级别的抽象,如规格说明,需求等;
- 识别、检索和修改值得重用软件/设计的简易手段;
- 鼓励重用的刺激手段;
- 共同事物引起的重用;
- 为缓解对软件可用资源需求的矛盾之大范围批量重用;
- 值得选择与采用的软部件;
- 更难的更高级抽象软件/设计重用;

需要的:

- 软件表示法,以表达技巧、约束、设计

策、基本理论及其有关信息(注意:在生存期前阶段的正向工程中,最好能得到,否则,在再工程期间,即使可能,但也将是困难的);

- 设计/代码的自建档(注意:别相信软件文档与代码相一致);
- 从设计生成代码,从而设计是代码的一种准确表示,积极的重用可以提高生产率并加深对问题域的了解;
- 特定应用领域的领域分析模型;
- 在所有抽象技术,需求技术和实现技术级上,基于本原的应用程序生成设施;
- 作为维护辅助的重用技术;
- 从现存代码抽象设计供新环境的端口系统用的手段;
- 识别部件的分析技术和修改部件适合新的应用的方法。

应调查研究的:

- 修复或替换决策;
- 库开发方法、公共化和检索机制;
- 鼓励重用/软件利旧的方法;
- 重用级别与重用成本间关系的经验研究。

二、转移管理

- 2.1 技术转让(略)
- 2.2 从研究到产品的转移(略)
- 2.3 支持转移的方法与工具(略)
- 2.4 CASE的使用与滥用(略)

三、小组与过程管理

- 3.1 支持小组工作的工具(略)
- 3.2 测度与测量(略)
- 3.3 质量(略)

四、可行的技术

- 4.1 元CASE技术(略)
- 4.2 库技术(略)
- 4.3 CASE 工具集成

正在进行的:

- 建立在公共窗口环境上的用户界面提供低

级别的表示集成。目前人们仍将注意工具的变化;

- 面向批处理的文件交换提供工具数据集成的部分解决办法;
- 框架结构正成功地支持公用工具的援用;
- 内部工具间通讯的直接界面和消息服务程序正提供控制集成的部分解决办法;
- 当前有一些成功的机制可用于文件级关系管理;
- 单个厂主或合伙人提供的私有工作台对狭窄的应用领域是成功的;
- 在许多情况下,工具买主或供货主正在从事扩充和组合现有工具的集成工作,以满足直接买主的需求;
- 在欧洲研究共同体,PCTE(可移植公共工具环境)提供了趋向一致的公共工具界面的定义。

尚未进行的:

- 尽管少数卖主提供了封闭式对象管理系统,但非卖品对象级(子文件)交换技术是不可用的;
- 部分集成的解决办法(仅满足单方面集成)并不能适应开发者的需要;
- 发展中的标准仍不成熟,不够强壮,也不能用实际产品予以很好地描述;
- 相对于现存工具,用户界面变化太大,甚至在同一GUI框架结构下的集成亦如此;
- 现有IRDS提出的标准仍不足以实现完全的工具体集成;
- 以私有库设施为基础的数据集成并不能适用于工具集成的目标;
- 只支持专业开发人员的环境并不能适应工具集成的目标。

需要的:

- 能够以保存语义内容的方式在工具间进行双向信息交换的能力;
- 通过大产业参与者大力承担义务/投入人力、开发原型、研究市场需求以及兼容产品(如OSF、PCTE),提高对标准的更大信任;
- 先积累工具集成方面的经验,最后定下标

准;

- 更好地同步卖主在市场上投放工具,以便提供集成工具间的连续兼容性;
- 具有灵活的(可扩充)元模型的工具集成库;
- 能支持强壮的软件工程过程控制模型和控制机制的工具集成设施;
- 与工作台无关的网络化异质集成框架结构;
- 全分布式数据集成设施(库);
- 在简单UNIX管道和过滤程序级别之上进行更复杂的内部工具通讯的标准;
- 对于CASE集成的参考模型的一致意见;
- 支持现有工具搬家和可扩充性的标准,以适应将来的工具和方法;
- 大型库的概念性控制方法和设施;
- 对于所有软件开发任务(技术、管理、支持)集成工具的能力;
- 所有的需要应与库相协调。

应调查研究的:

- 涉及整个生存期的少数标准元模型;
- 探讨元模型定义的技巧(元-元模型);
- 正向和逆向工程的集成;
- 工具环境建造设施(软件工厂);
- 用软件工程过程模型指导工具的集成;
- 如何建立有效的库导航设施;
- 用以测量集成有效性的软件产品和过程的测度;
- 最佳的库事务协议;
- 工具数据语义、服务和工具行为的形式定义;
- 如何建立与策略独立的工具服务,如函数不应依赖于存取协议;
- 怎样构造工具和集成框架结构对其它工具的影响最少以及能用最少量知识进行工具使用、援用和存取(“插件式工具与服务”);
- CASE对其硬件/软件环境需求的框架结构;
- 对支持集成化CASE环境的硬件/软件需

求的更好理解。

五、方法和工具

5.1 需求的引出与可跟踪性(略)

5.2 重用

为讨论起见,采用下述重用的现用定义:重用指生存期中产生的人工制品(产品或过程)在另外的上下文中的再应用。

尚未进行的:

- 在分类、选择、理解、修正和适应性等方面,现行CASE技术并未对软件重用提供适当的支持。

需要的:

- 提供一个可重用部件库是不够的;还需要有支持重用部件的分类、选择、理解和适应性的方法学;
- 为重用提供的支持不仅涉及程序部件的重用,而且应包括那些生产出来的任何人工制品,如设计规格说明和测试数据;
- 协作支持:
 - 在重用前检索一个要求适应的大型部件,可能需要小组的协作及合力支持;
 - 可重用性的实现取决于有组织地共享小组间的经验与知识以及应用领域;
 - 在多用户搜索相同或相似的候选功能情况下,小组协作可以产生更有效的可重用集。
- 规格说明级的重用既需要说明应用的方法知识,也需要其领域知识;
- 为了指导潜在的“重用者”,可以使用智能辅助工具来把握开发规格说明的过程;
- 可执行规格说明实际上增加了该规格说明的有效重用;
- 部件库支持应包括:
 - 获取或检索生存期各种人工制品的能力;
 - 从现存应用提取可重用部件的能力;
 - 部件使用的可跟踪性,因此可以区分更正或增强。

- 潜在的可重用部件(如设计历史)必须具体化,即必须有一种表示法,用以获取该部件的实例;
- 为建立可重用部件库所提供的支持应包括设计与捕获重用部件的能力;
- 重用过程部分包括该过程的监控,以列入诸如被重用部件、被重用次数之类信息,以便改进重用过程;
- 适应或取舍:
 - 重用系统应适应于一个组织并且为它所适应;
 - 重用系统对于应用领域应是可取舍的;
 - 重用系统与现存组织的过程应是相容的,并且能适应递增或渐进的转移;
 - 即使有了重用系统的可适应性,但仍需要一种方法来保证基本重用过程的完整性。
- 重用系统应提供一个信息库,通过利用下列知识,支持应用的开发与维护:
 - 应用领域;
 - 开发的方法与模型;
 - 设计历史。
- 应调查研究的:
 - 寻找对非代码部件使用的表示;这些部件包括过程和产品;
 - 调查研究基于机器的学习和基于情况的推理,以作为设计重用工具支持的技术。

5.3 形式方法

正在进行的:

- 基于文式(paper based)的系统规格说明语言;
- 有限领域(如通讯协议)中的规格说明语言的支持工具;
- 大部分生存期中的一般方法的应用。

尚未进行的:

- 对任一规格说明语言的全面工具支持;这可能包括基于知识的帮助、证明辅助等;
- 迅速扩大到产业规模(而不是规格说明);
- 从非形式化方法到形式化方法的转换基

础;

- 理解有效性的技巧(规格说明确认);
- 规格说明与系统间的联系(可跟踪性);
- 无需适当教育的结构化自然语言的使用;
- 形式方法的用户友好界面;
- 产生和使用形式表示技巧的形式过程;
- 形式上可信的验证式设计(人们喜欢测试)。

需要的:

- 更好的形式规格说明语言,包括并发规格说明;
- 用规格说明语言开发的优良工具和支持结构;
- 好的形式语言的用户界面;
- (转换到其它语言的)生产强度及原型执行;
- 使用通用方法的变换工具和将它们变换成形式方法的途径;
- 规格说明的推理(以辅助确认);
- 考察设计空间以选择可供选择的方案;
- 设法避开实现特有的特性;
- 大规模实例研究;
- 可跟踪性;
- 更好地教育学会形式方法;
- 半结构化规格说明的验证。

应调查研究的:

- 现行规格说明方法的一般化特性:适应性、可用性;
- 如何论证使用形式方法的收效(包括形式定义的系统有效性);
- 公司选用形式方法的需要;
- 帮助利用形式方法的核心CASE技术;
- 有良好基础的形式理论与工具使用的结合;
- 工具支持与语言表达能力间的折衷;
- 形式规格说明中语言理论的使用,例如支持命名;
- 表征形式规格说明部分的能力;
- 用户需求与形式规格说明的其它方面;
- 规格说明中实现相关与实现无关;

5.4 面向对象(O-O)方法

正在进行的:

- 结构化分析和O-O分析实际上可以输入O-O设计阶段;
- O-O程序设计的某种支持(语言工具);
- 某些单阶段(单个活动)工具;“点解(Point Solutions)” (例如, OOA, C++程序设计, HOOD, Ada前端工具);
- 用户界面工具箱和工具(如, CASE:PM, EASEL, NeXT Step, Choreographer);
- 以C++为中心的工具体开发;
- HOOD的CASE支持(特别在欧洲);
- 有限的模拟支持(如, MODSIM);
- 使用O-O工具建立原型;
- 可重用类库和框架结构(推向设计级的重用)。

尚未进行的:

- CASE未使O-O系统十分易于重用;
- O-O技术的生存期支持未达到完全集成;
- 技术和方法的集中(如, 语言、数据库和O-O设计);
- 支持甚大规模O-O系统(指规模与复杂性)的工具和方法;
- 重用或维护中的普遍增益;
- 现有O-O方法的全支持。

需要的:

- O-O CASE工具应支持需求、与实现无关的(设计)域和结构间的变换;
- O-O程序设计活动与O-O分析和设计的充分集成;
- 开发部件(包括设计)重用的显式支持;
- 大型或小型O-O系统的程序设计支持;
- 现行演进与维护的显示支持;
- 混合型风范开发的支持(如, ERA与

OOA),

- 多种技术O-O开发的支持(如, 语言、数据库、分析、设计);
- 信息隐藏的特别支持和实施;
- 对象类定义的测试实例自动生成;
- 程序设计支持的更强壮环境(如, 劝阻胡乱删改);
- 以技术和方法为中心的市场;
- 领域分析的支持;
- O-O数据库分析与设计工具。

应调查研究的:

- 什么类型的活动和系统最适合于O-O CASE工具?
- O-O能用于传统的MIS应用和系统吗?
- CASE和O-O怎样结合才能支持重用和维护?
- CASE对下列各项怎样提供帮助?
 - 对O-O系统的性能和并发性的说明;
 - 执行和模拟O-O系统;
 - 形式化O-O方法的集成;
 - O-O开发项目的项目估计、计划和控制。
- CASE怎样才能帮助集成各种各样的O-O技术和活动?
- O-O系统与类库的配置管理;
- O-O过程中可跟踪性的CASE支持;
- 断言、前件和后件的CASE支持;
- 怎样支持超大型O-O开发;
- O-O系统协作开发的支持;
- 分解运算(factoring)的工具及其标准。

5.5 验证和确认工具(略)

[徐仁佐 刘莲君 编译自《Proceedings of 13th ICSE, Austin, Texas, USA, May 1991]