



# 2024 AI+研发数字峰会

AI+ Development Digital summit

AI驱动研发变革 促进企业降本增效

北京站 08/16-17

## 大语言模型时代需求自动追踪技术

匡宏宇 南京大学



## 匡宏宇

CCF软件工程专委会执行委员/持续研究自动化软件追踪十余年

南京大学软件学院助理研究员，软件研发效能实验室核心成员，十余年来持续研究软件可追踪性、代码库挖掘、程序理解等软件开发相关方向，曾获ACM南京分会优秀博士论文、AIIDE 2019 Best Paper、教育部-华为“智能基座”栋梁之师等奖项，《开发运维一体化成熟度模型》国家标准的主要起草人之一（产品与需求部分），共发表论文20余篇，其中一作/通讯作者论文11篇（包含ICSE、FSE、ASE等顶尖软工会议论文），应邀担任过SANER 2024 程序委员，JSEP、JSS、SoSyM期刊审稿人，EASE 2024 Publicity Lead (China)等学术兼职，主持南京大学软件学院需求课程体系建设与教学工作，主持并结题国自然青年基金一项，参与并完成多项重点企业合作。

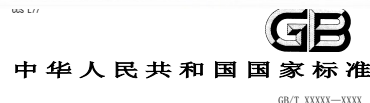


# ► 软件研发效能@南京大学

- DevOps/研发效能全方位科研
- DevOps教材及**课程**建设
- DevOps知识体系构建
- DevOps能力成熟度**国家标准**制定
- DevOps/研发效能全流程**产业**合作研究
- DevOps/研发效能中国**年度调查**
- DevOps/研发效能中国**社区**建设
- DevOps/研发效能国内/国际**学术**论坛
- DevOps/研发效能学术**论文**(260+)及**专利**(80+)
- CCF-华为胡杨林基金成立以来**唯一连续3年持续获得资助**的科研团队



CCF-华为胡杨林基金资助。



系统与软件工程 开发运维一体化 能力成熟度模型

System and Software Engineering - Development and Operations - Capability Maturity Model

(报批稿)

在提交反馈意见时，请提供知道的相关专利支持性文件一并附上。

XXXX - XX - XX 发布

XXXX - XX - XX 实施

国家市场监督管理总局  
国家标准化管理委员会 发布

GB/T XXXXX-XXXX

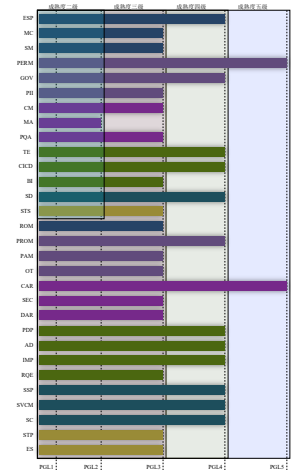
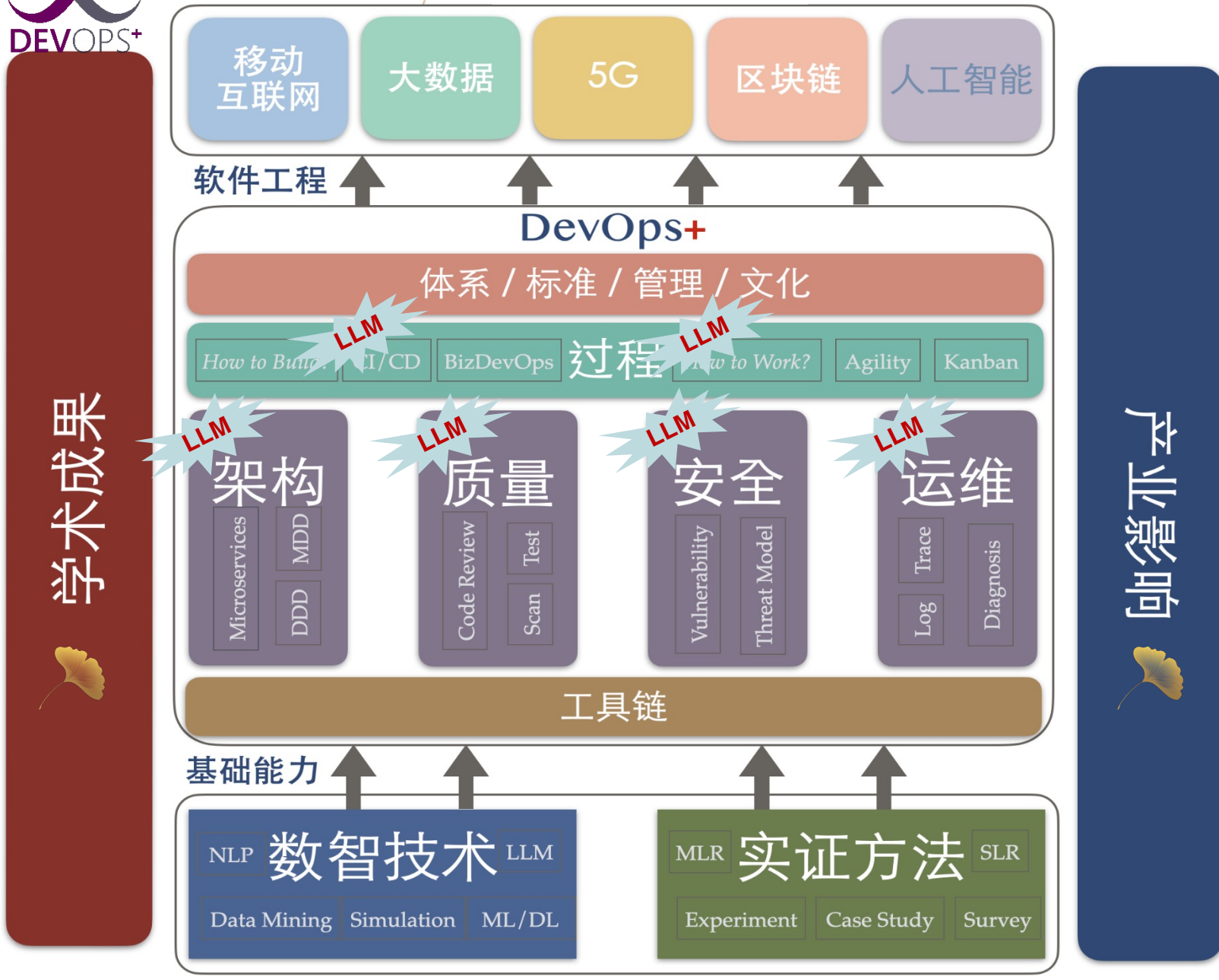


图1 开发运维一体化能力成熟度模型





# 目录

## CONTENTS

1. 需求自动追踪技术背景与发展
2. LLM时代需求追踪的重要价值
3. 基于LLM的需求追踪技术初探
4. 本人代表研究简介（请多指正）
5. 总结与展望

# PART 01

## 需求自动追踪技术背景与发展



# ▶ 引言：为何要在软件项目开发中“溯源需求”？

## • 项目开发者视角：如何快速融入团队？以下为亲身经历

- 被某世界级通讯技术公司专家当面质问：“你是大学老师？你们课上都在教些什么？？”
  - 大哥为何愤怒：招来的名校**新人程序员**“不指定具体PRD和单元测试就写不出代码”
- 假设我们的培养“完美无缺”：学生充分掌握数学基础、编程范式、主流框架、实践经验
  - 但TA大概率仍无法全面快速“上手”主流项目：**不了解（复杂）项目的系统功能与演进历史**
  - 氛围良好、成熟稳定的开发团队可以缓解这一问题，但**问题本质是系统功能与实现之间缺乏明确关联**



## • 项目管理者视角：如何治理、盘活已有项目资产？

- 知名网游IP《魔兽世界》：“原汁原味”的怀旧服 vs. “锐意创新”的私服
  - 掌握全部项目资产的官方 – 直接上线包含所有缺点的经典版本
  - 自行破解还原的“同人” – 推出多个广受好评的改版（官方对此类明确侵权行为态度暧昧）
- 为何？ - 项目**代码复杂冗余**，开发团队**人员流失**，项目**退化为遗产代码（只能运行，无法维护更新）**
  - 旁证：永远**16格**的初始背包容量，官方解释“**相关代码在整个游戏最底层，无法精准定位，贸然修改将导致无数BUG**”
  - 2018年7.3.5版本尝试升级为**20格**，结果导致该版本在**地图、进度、角色、装备**等各个方面均涌现出严重错误



如何系统性应对此类问题？

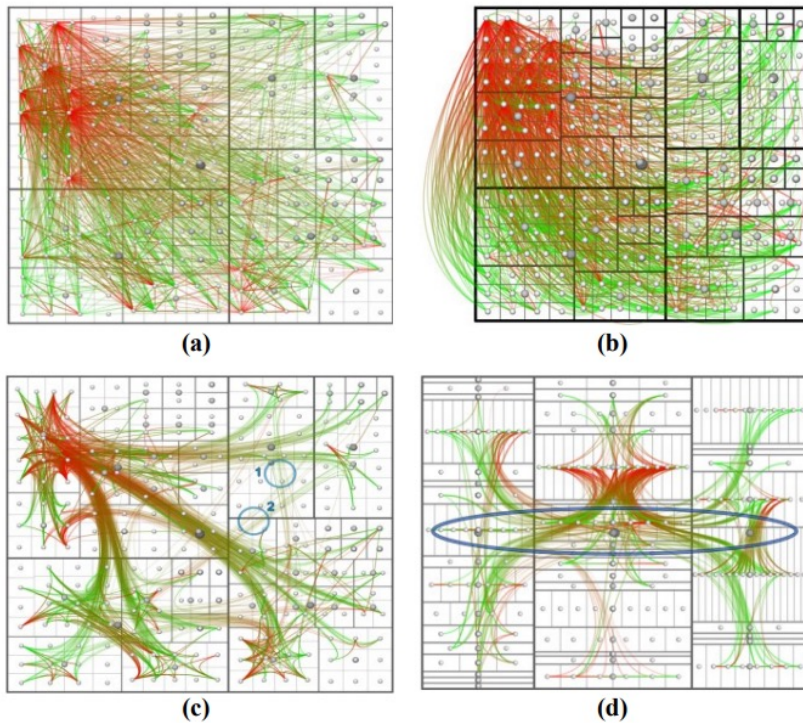
# ► 需求可追踪性：显式标识需求以及其它软件制品之间的联系

- 软件可追踪性是软件系统所能具有的一种特性：

[CoEST, the Center of Excellence for Software and Systems Traceability,  
<http://www.coest.org>]

- 系统中任意一个可被唯一标识出的**软件制品** (*software artifact*, 如需求、设计、代码、测试等) 与其它制品之间的**关联关系**

- 这些关联关系能够**随时间的变化而得到维护**
- 由这些关联关系组成的网络可以解答与一个软件系统及其开发流程相关的各类问题
  - 最早以**需求可追踪性**的方式提出[Gotel@RE 1994]
  - **重点关注需求** (功能期望) **到代码** (实际运行) **可追踪性**



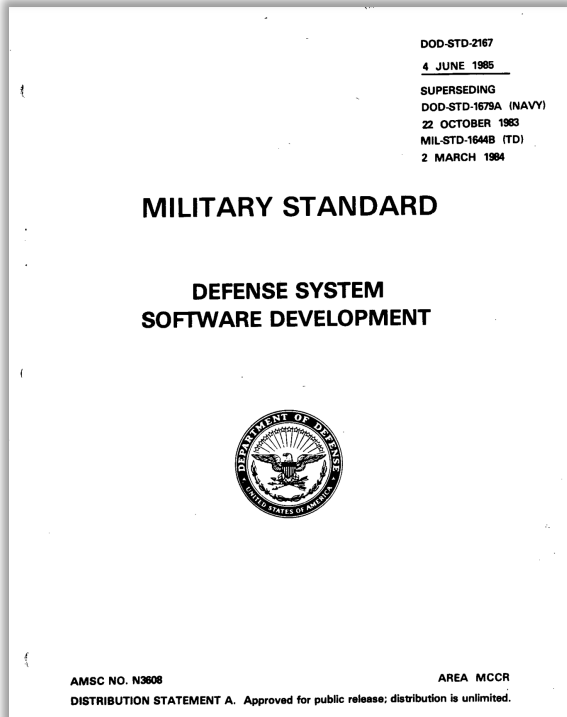
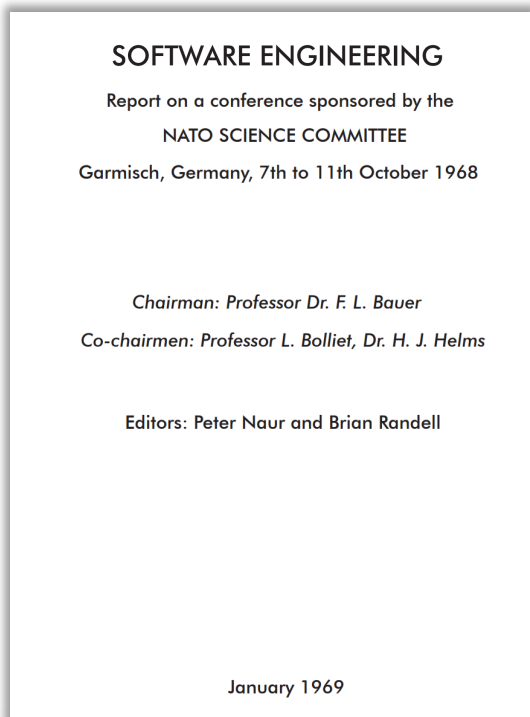
Requirement Trace Matrix (RTM)

	R0	R2	R5
VODClient.init()	X		
ListFrame.buttonControl3_actionPerformed()	X	X	
ListFrame()	X	X	
ServerReq.getmovie()	X	X	X
ListFrameListener3.actionPerformed()		X	X





# 需求可追踪性的诞生



发生了什么导致需求可追踪性“出圈”  
— 从军用项目走向一般软件开发？



## 1968+：概念探索

- 在1969年NATO 软件工程会议（定义了著名的“软件危机”问题）上讨论了软件可追踪性的概念与作用

## 1970s-1980s：开发实践

- 70年代军用软件强制要求，美国防部1985 年发布DOD-STD-2167A标准
- IEEE Std 830-1984: IEEE Guide to Software Requirements Specifications

## 1994+：正式定义

- Gotel 等人正式定义了需求可追踪性，提出需求可追踪性应该在软件系统中“普遍存在” [Gotel@RE1994]

# ▶ 需求可追踪性成为软件开发规范

- 诱因：“The Patriot Missile Failure” ...的快速修复
  - 爱国者导弹拦截失败导致的海湾战争中美军单次最大损失
  - 原因：战时长期开机积累的时钟漂移（使用的浮点数精度不足）
  - 1991.2.25日袭击发生，2.26日制造商雷神发布修正软件
- 结果：可追踪性引发热议并在90年代成为软件开发规范
  - “You cannot manage what you cannot trace”  
[Watkins@IEEE Software 1994]
  - 需求可追踪性应该在软件系统中 “普遍存在” [Gotel@RE1994]
  - 软件团队能力成熟度CMMI 3级以上 “要求实现” 可追踪性
    - 仍是当前诸多关键领域软件的强制要求，如DoD、FAA、FDA、NASA等机构、核心网络设备、金融企业的safe-critical软件

## The Patriot Missile Failure

**The Patriot Missile Failure** On February 25, 1991, during the Gulf War, an American Patriot Missile battery in **Dharan, Saudi Arabia** failed to track and intercept an incoming Iraqi Scud missile. The Scud struck an American Army barracks, killing 28 soldiers and injuring around 100 other people. A report of the General Accounting office,

The Patriot Missile Failure - University of Minnesota  
[www-users.cse.umn.edu/~arnold/disasters/patriot.html](http://www-users.cse.umn.edu/~arnold/disasters/patriot.html)

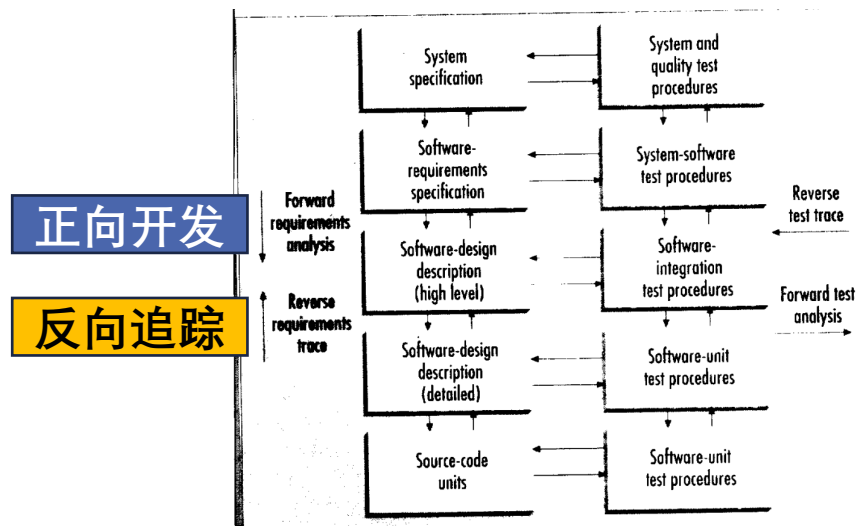


Figure 2. How requirements are analyzed (forward) and traced (reverse) and tests traced (reverse) and analyzed (forward) within the documentation hierarchy.



# ► 需求可追踪性走入日常开发的价值、困难与挑战

- **价值：有效提升（敏捷）软件开发的质量与效率**
  - **（普通新手）开发者通过软件可追踪性可获取（至少）以下关键信息：**代码库中需更改的代码元素，受影响的 user story 与测试用例，以及可咨询的项目组内开发者 [Cleland-Huang@FOSE 2014]
    - **有效帮助理解已有代码与开发任务：**“当前软件开发人员阅读代码与编写代码的耗时比例超过10:1”
      - Robert C. Martin 2008 - 《Clean Code: A Handbook of Agile Software Craftsmanship》
  - **软件维护任务**在需求到代码可追踪性的支持下，**正确率提高了60%，效率提高了20%** [Maeder@ICSM 2012]
  - 在质量保证[Rath@TSE 2017]、系统安全[Moran@ICSE 2020]、缺陷定位[Niu@ICSE 2023]等研究中被广泛应用
- **困难：人工实现可追踪性代价大、成本高**
  - **人工追踪成本近似于开发本身**，导致现实系统中并不常见[Ramesh@Journal of Defense Software Engineering 1995]
- **挑战：如何使得需求（软件）可追踪性“广泛存在”**
  - Grand Challenge of Software Traceability: Ubiquitous [Gotel@RE 2012]
  - 引入自动化分析技术，**构建高效、可信的需求自动追踪**技术与工具



# 自动化需求追踪在软件工程领域研究中的发展

Workshop

**3rd International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE 2005)**

Jonathan I. Maletic  
Department of Computer Science  
Kent State University  
Kent Ohio 44242 USA

Giuliano Antoniol  
Department of Engineering  
University of Sannio  
Benevento, Italy &  
Ecole Polytechnique  
de Montréal,  
Montréal, QC, Canada

Jane Cleland-Huang  
School of Computer Science,  
Telecommunications,  
and Info Systems  
DePaul University  
Chicago Illinois  
USA

Jane Huffman Hayes  
Department of Computer Science  
University of Kentucky  
Lexington, Kentucky  
USA

Establishing and maintaining traceability links and consistency between software artifacts produced or modified in the software life-cycle are costly and tedious activities that are crucial but frequently neglected in practice. Traceability between the free text documentation associated with the development and maintenance cycle of a software system and its source code are crucial in a number of tasks such as program comprehension, software maintenance, and software verification & validation. Finally, maintaining traceability links between subsequent releases of a software system is important for evaluating relative source code deltas, highlighting effort/code variation inconsistencies, and assessing the change history.

The main theme of the workshop is focused on understanding and defining the foundations for consistency and change management of software systems within the scope of artifact-to-artifact (model-to-model) traceability.

The workshop will address the following issues:

- A formal definition of model to model traceability
- Traceability between artifacts and processes
- The semantics of traceability links
- Recovery of traceability links
- Visualization of traceability links
- Interoperable approaches to support traceability
- Traceability in emerging forms of software engineering including production lines, frameworks, components, etc.

The goals of the workshop are to:

- Broaden awareness within the software engineering community of the potential for the application of traceability
- Facilitate the exchange of ideas and interaction between international researchers
- Define open research problems faced in realizing usable approaches for traceability
- Construct a foundation of materials for future research on traceability

For more information please visit the workshop web site is: <http://te.cs.depaul.edu/tefse05/>. The workshop proceedings are available through the ACM digital library.

Copyright is held by the author(s).  
ASE '05, November 7-11, 2005, Long Beach, California, USA.  
ACM 1-58113-995-4/05/0011.

**Software Traceability: Trends and Future Directions**

Jane Cleland-Huang  
DePaul University  
SAREC  
Chicago, IL, USA  
jhuang@cs.depaul.edu

Oriana C. Z. Gotel  
Department of Computing  
University of Kentucky  
New York, NY, USA  
olly@gotel.net

Jane Huffman Hayes  
Department of Computing  
University of Kentucky  
Lexington, KY, USA  
hayes@cs.uky.edu

Patrick Mäder  
Technische Universität Ilmenau  
Software Systems Group  
Ilmenau, Germany  
patrick.maeder@tu-ilmenau.de

Andrea Zisman  
The Open University  
Department of Computing  
Milton Keynes, UK  
andrea.zisman@open.ac.uk

**ABSTRACT**

Software traceability is a sought-after, yet often elusive quality in software-intensive systems. Required in safety-critical systems by many certifying bodies, such as the USA Federal Aviation Authority, software traceability is an essential element of the software development process. In practice, traceability is often conducted in an ad-hoc, after-the-fact manner and, therefore, its benefits are not always fully realized. Over the past decade, researchers have focused on specific areas of the traceability problem, developing more sophisticated tooling, promoting strategic planning, applying information retrieval techniques capable of semi-automating the trace creation and maintenance process, developing new trace query languages and visualization techniques that use trace links, and applying traceability in specific domains such as Model Driven Development, product line systems, and agile project environments. In this paper, we build upon a prior body of work to highlight the state-of-the-art in software traceability, and to present compelling areas of research that need to be addressed.

**Categories and Subject Descriptors**

D.2.1 [Software Engineering]: Requirements/Specifications

**General Terms**

Documentation

**Keywords**

Software traceability; road map

**1. INTRODUCTION**

Software traceability has long been recognized as an important quality of a well-engineered software system [37].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided the copies are not made or distributed for profit or commercial advantage and the copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Notwithstanding to rights reserved, no part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or by any information storage and retrieval system, without permission from ACM Publications Department.

FOSE '14, May 31 - June 7, 2014, Hyderabad, India  
ACM 978-1-4503-2862-4/14/05  
<http://doi.org/10.1145/2593882.2593891>


78]. Defined by the Center of Excellence for Software and Systems Traceability (CoEST) as "the ability to interrelate any uniquely identifiable software engineering artifact to any other, maintain required links over time, and use the resulting network to answer questions of both the software product and its development process" [14], traceability is a required component of the approval and certification process in most safety-critical systems. For example, the DO-178C standard [73], which the USA Federal Aviation Administration (FAA) has established as the means of certifying that software aspects of airborne systems comply with airworthiness requirements, specifies a very detailed set of traceability requirements including the need to provide "traceability between source code and low-level requirements" in order to "enable verification of the absence of undocumented source code and verification of the complete implementation of the low-level requirements." Similarly, the USA Food and Drug Administration (FDA) states that traceability analysis must be used to verify that the software design implements the specified software requirements, that all aspects of the design are traceable to software requirements, and that all code is linked to established specifications and test procedures [30].

Despite its importance, traceability is perhaps one of the most elusive qualities of the software development process. The cost, effort, and discipline needed to create and maintain trace links in a rapidly evolving software system can be extremely high. Moreover, its benefits often go unrealized in practice, either due to ill-defined and ad-hoc traceability processes, poor user training, or a lack of effective tooling [37, 5]. Reflecting this state of practice, the "Critical Code: Software Productivity for Defense" report commissioned by the U.S. Department of Defense identified requirements traceability as one of the seven technology areas on which research should be targeted in order to assure the safe and correct operation of current and future software intensive systems [15].

In this paper, we set out a focused research agenda for software traceability. We build upon prior work that first identified the Grand Challenge of Traceability [35] and then proposed a high-level roadmap for achieving it [34]. Motivated by current practice, this paper draws out and drills down on the key areas in which research focus is needed. It details the state of the art in each of these areas and then focuses attention on specific high priority research needs.

**Grand Challenges of Traceability: The Next Ten Years**

Giuliano Antoniol, Jane Cleland-Huang, Jane Huffman Hayes, and Michael Vierhausner



## 2005：确立方向

- 领域内专业workshop—TEFSE开始接受软件可追踪性相关主题学术投稿

## 2014：承上启下

- 总结软件可追踪性的发展趋势并展望未来方向：满足实际需求，实现可追踪性的低成本高价值、可配置、可信、可控、便捷

## 2017：继往开来

- 探讨下一个十年的发展：如何利用软件代码库中的大数据、基于（深度）学习的自动化追踪技术、生产实践中的运用等



# 近15年需求自动追踪生成技术发展文献综述 (2010-2024)

搜索词条:

(traceability OR "requirements trac\*" OR "software trac\*" OR "link recover\*" OR "trace recover\*")

AND (artifact OR artefact OR document OR software OR requirement OR issue)

AND (recover\* OR retriev\* OR generat\* OR creat\*)

NOT (food OR "network security" OR "sensors network" OR "supply chain" OR "genetic traceability" OR "block chain")

文献来源:

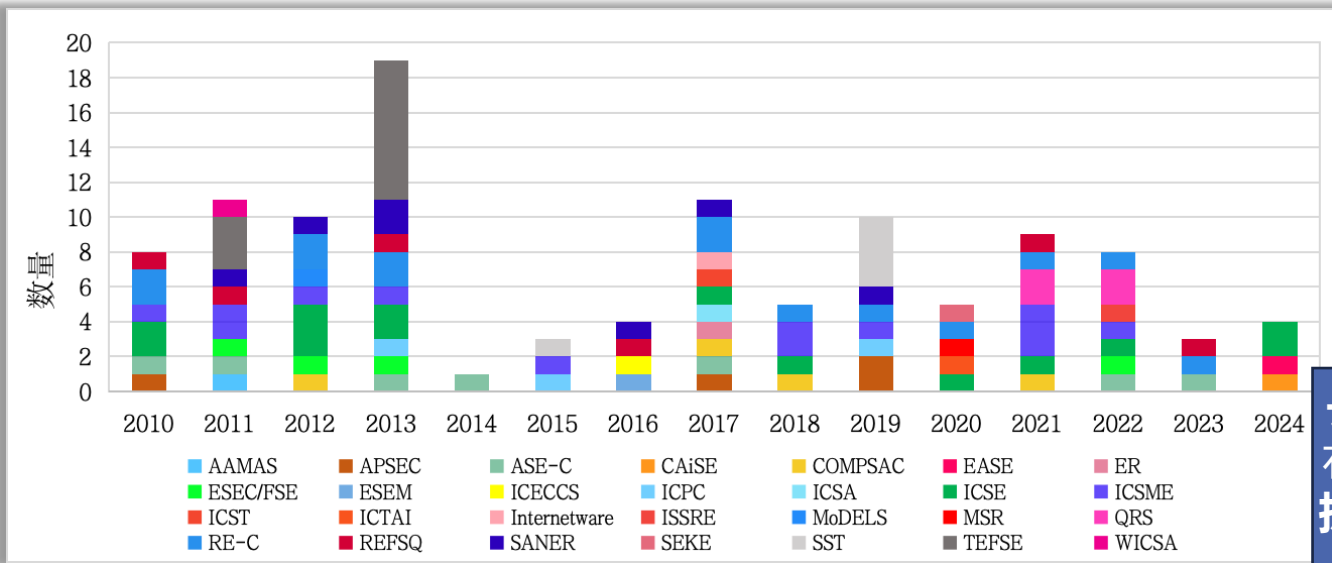
数据库名称	搜索范围	初步检索结果
ACM Digital Library	标题 + 摘要 + 关键词	1969
IEEE Xplore	标题 + 摘要 + 关键词	799
Web of Science	标题 + 关键词	1910
Science Direct	标题 + 摘要 + 关键词	54
SpringerLink	标题 + 摘要 + 关键词	3488
Scopus	标题 + 摘要 + 关键词	1781

筛选条件:

	包含
包含标准1	论文发表时间: 2010.1–2024.6之间.
包含标准2	论文相关性: 主题与软件可追踪相关.
包含标准3	论文完整性: 如会议论文,进一步扩展为期刊论文, 只保留期刊论文版本, 排除会议论文版本
包含标准4	论文权威性: 经过同行评审且已正式发表的论文, 排除书籍章节、硕博论文等
包含标准5	论文语言: 用英语撰写的论文.
包含标准6	论文页数: 大于等于5页
包含标准7	论文等级: CCF A/B/C, 或可追踪相关会议TEFSE, SST
	剔除
剔除标准1	违反包含标准中的任意一条
剔除标准2	论文主题为软件可追踪的维护、管理、使用与应用, 而非可追踪生成技术

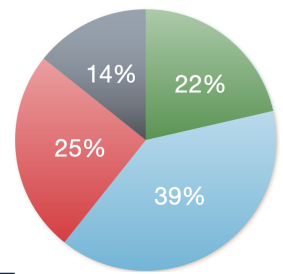
# 相关文献在高水平会议期刊上的分布

会议



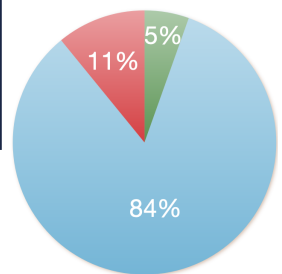
会议111 (67%)      期刊55 (33%)

会议



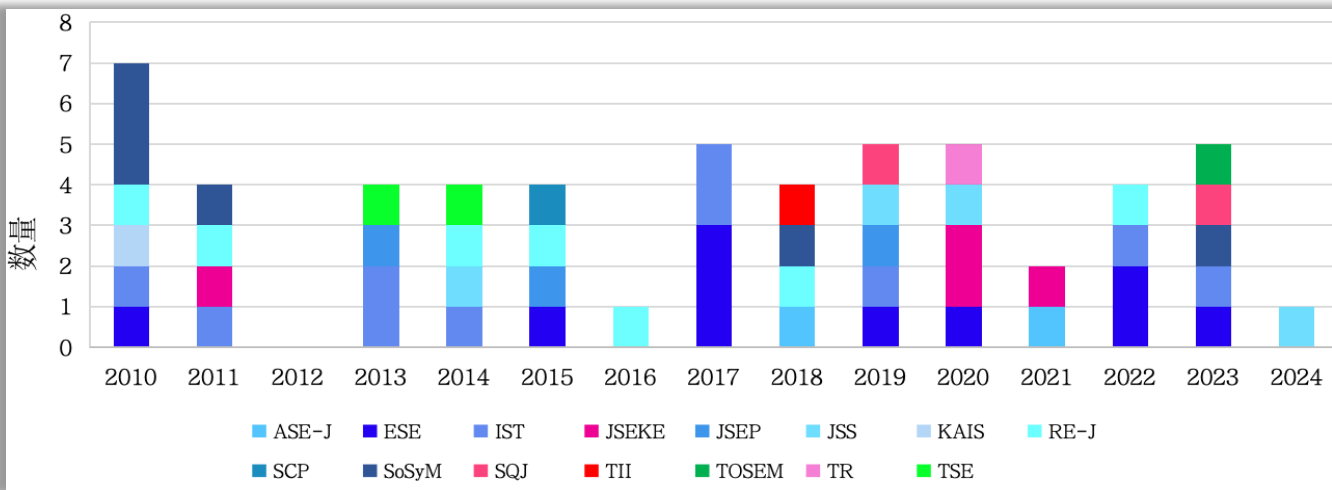
长期研究热点，重点分布于顶会ICSE、软件维护主题的一流会议期刊、专题Workshop等

期刊

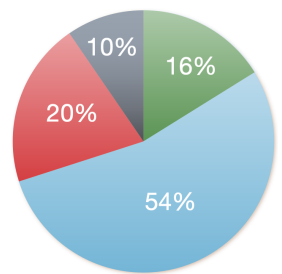


CCF-A  
CCF-B  
CCF-C  
其他

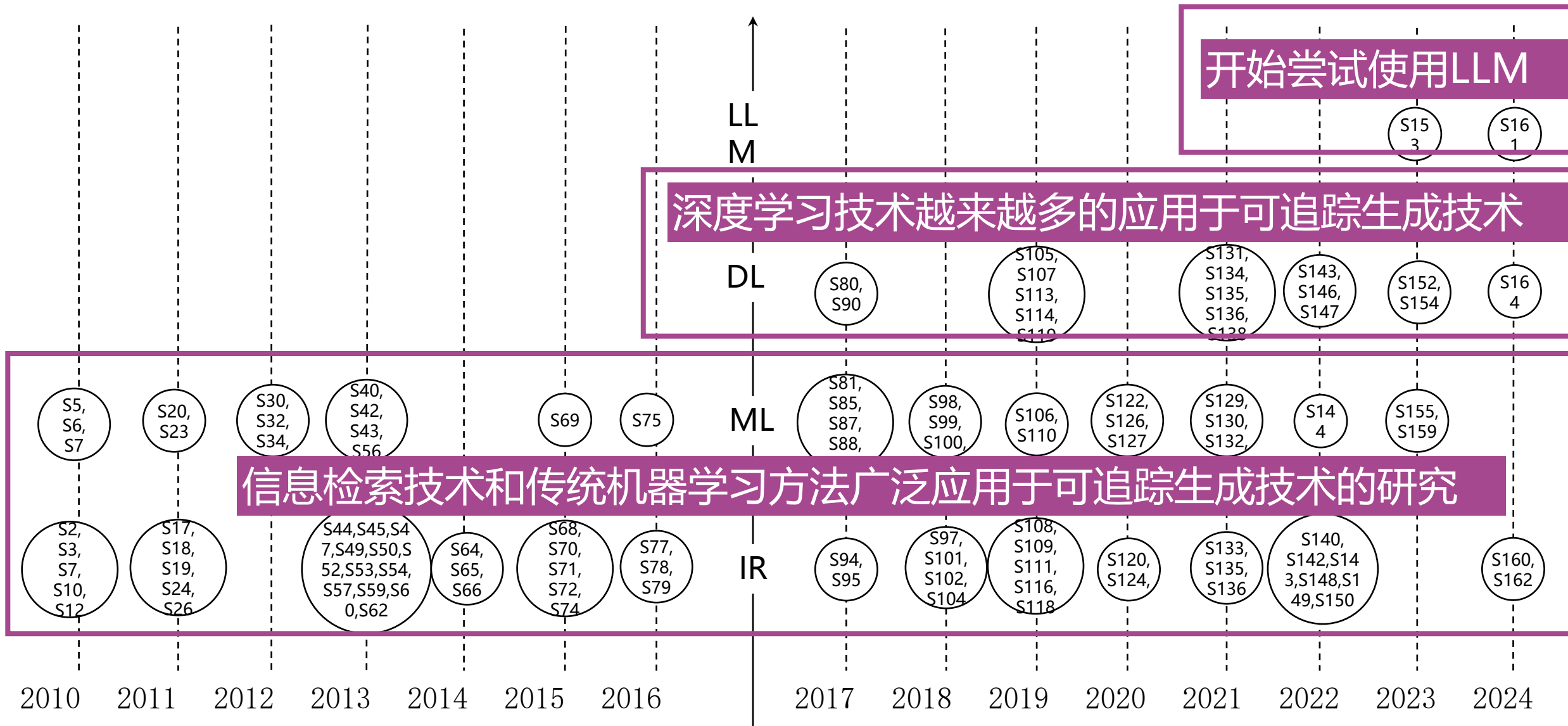
期刊



会议+期刊

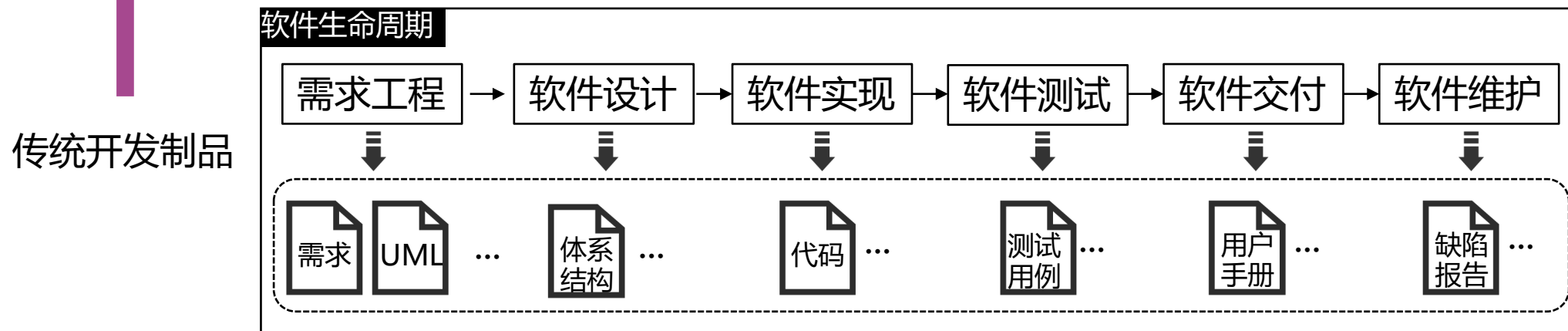
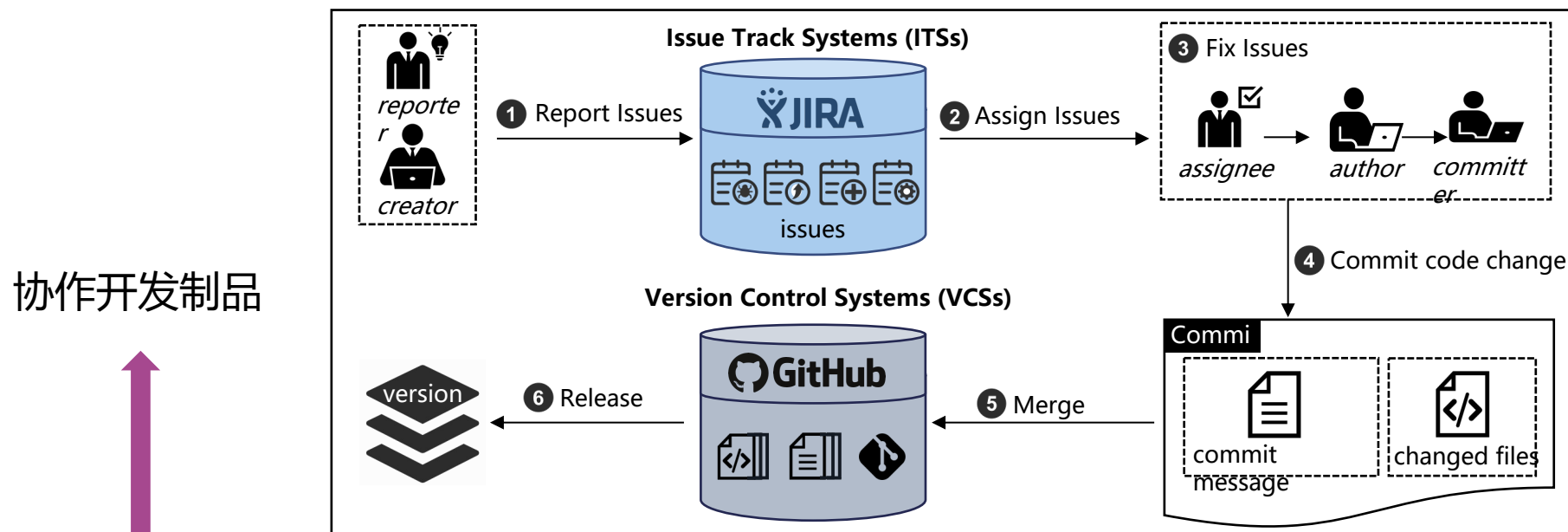


# 自动追踪技术演化趋势

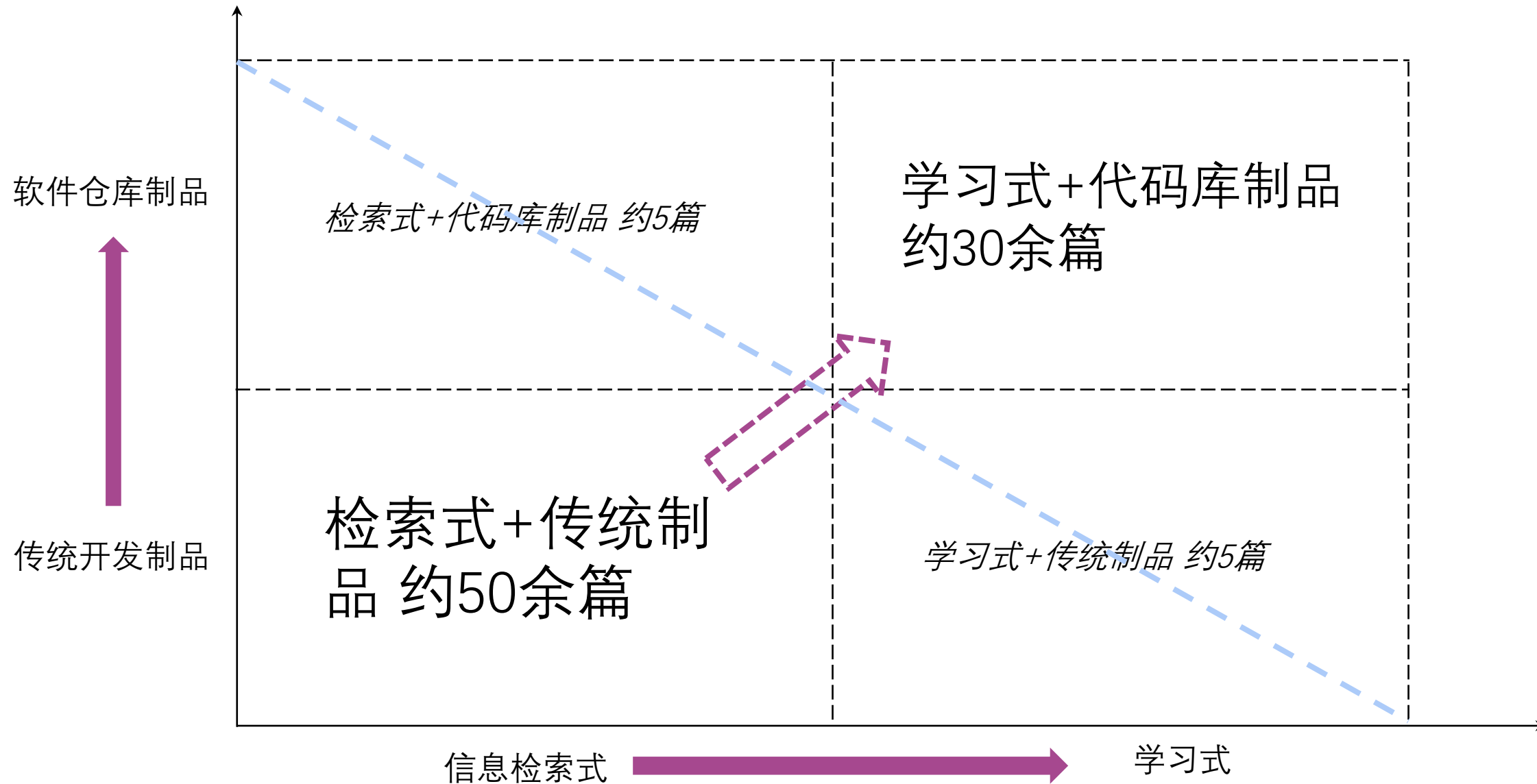




# 自动追踪目标制品：从传统文档到代码库制品



## LLM + 需求自动追踪后续工作展望



## PART 02

# LLM时代需求追踪的重要价值



# ▶ 需求可追踪性对软件开发的积极作用 – 知名软件产品视角

- 某著名蓝色支付软件 – 安卓APP “瘦身”
  - 非必须系统功能的遗留使得软件过于 “臃肿” ，导致用户体验下降并增加安全风险
  - 采用动态监控的方式逐步锁定冗余功能
    - “MiniMon: Minimizing Android Applications with Intelligent Monitoring-Based Debloating” , ICSE 2024
  - 若有可追踪性的支持，可在源码层面实现功能定位与系统重构
- 某造车大成功的IoT生态公司 – 安卓OS社区被迫封闭
  - 机型丰富加重 “安卓碎片化” ，被迫大幅弱化系统的用户可定制性，关键的12.5版本发布也被拖累
  - 若有可追踪性的支持，可在源码层面实现 “自底向上” 的软件产品线模型构建与系统特性定制
- 大模型出现后的时代需要 – 开发过程数据资产化是大模型助力软件开发的前提
  - 要充分利用大模型的对话式交互与超强文本归纳能力，必须有效治理、组织、利用开发过程中的各类数据
  - 需求可追踪性能够在不同类型、不同格式的制品之间显式建立追踪线索，是开发数据资产化的关键



# ▶ 需求可追踪性在大模型时代对软件开发的重要作用

## • 目前LLM是否已彻底改变软件开发？

- 对程序员的个体编码能力提升较为明显
- 但从软件开发的人、过程、制品三个维度来看，整体效率并未大幅提升 – “理解代码”的难度要远高于“写出代码”
  - 基于系统功能显式关联各个软件制品的软件（需求）可追踪性是程序理解的关键

## • 如何更好地在特定领域项目开发上应用LLM？

- ChatGPT类通用大模型是在海量数据上的一次“最优化拟合”，难以直接回答特定问题，必须依赖有效的Prompt引导通用大模型“聚焦”
- 进一步通过开发数据资产化积累项目日常开发的高质量数据，从而基于这些数据构建专用的“小”大模型
  - 数据资产化的核心是软件可追踪性的高效实现

## • 将软件开发与LLM充分结合后的前景展望（代差）？

- 有效应对软件开发的进一步复杂化
  - LLM时代会出现更多AI-Intensive系统，代码将不再是唯一关键制品（额外涉及AI模型与训练数据）
    - 对软件（需求）可追踪性也提出了更高要求
- 开发效率持续提升后增强企业业务创新能力BizDevOps

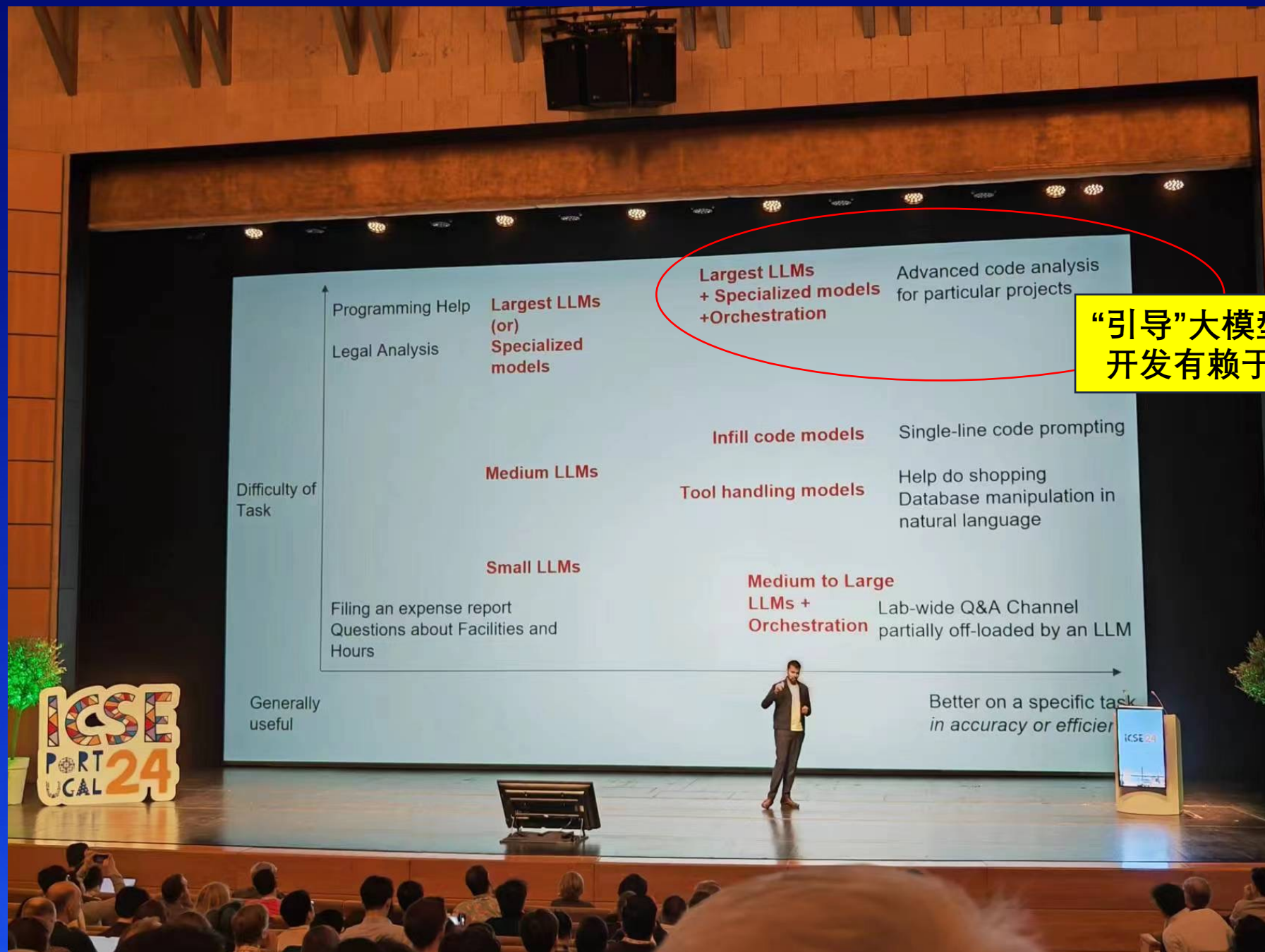
## Empirical Evaluation of ChatGPT on Requirements Information Retrieval Under Zero-Shot Setting

Jianzhang Zhang<sup>1</sup> | Yiyang Chen<sup>1</sup> | Nan Niu<sup>2</sup> |  
Yinglin Wang<sup>3</sup> | Chuang Liu<sup>1</sup>

1. Prompt learning for specific requirements retrieval and other NLP4RE tasks could be studied to more sufficiently utilize the domain knowledge encoded by general LLMs.
2. It is necessary to devise RE domain LLMs based on open source general LLMs with RE domain datasets and instructs to further improve the performance of LLMs on various NLP4RE tasks.
3. To better support the above two research directions, it is also essential to build high-quality benchmark requirements datasets for training and comprehensively evaluating the devised RE domain LLMs.
4. It is also valuable to investigate how to combine RE domain LLMs and the formal methods to better support requirements modeling and verification efficiently.
5. RE practitioners, e.g., analysts, could further fine-tune the RE domain LLMs with their own requirements which are usually business confidentiality. The fine-tuned models can provide requirements analysis assistance with an chat-based and end-to-end manner.

1. 大模型在具体开发任务上有赖于对专用数据的利用
2. 有必要在通用大模型的基础上构建专用大模型
3. 构建专用大模型需要构建高质量数据集与验证集
4. 将大模型与已有技术结合是值得去做的宝贵尝试
5. 项目专家依然需要用自己项目微调专用大模型，微调后的模型可以提供对话式的端到端交互





“引导”大模型有效助力特定软件项目开发有赖于可追踪性的实现与引入

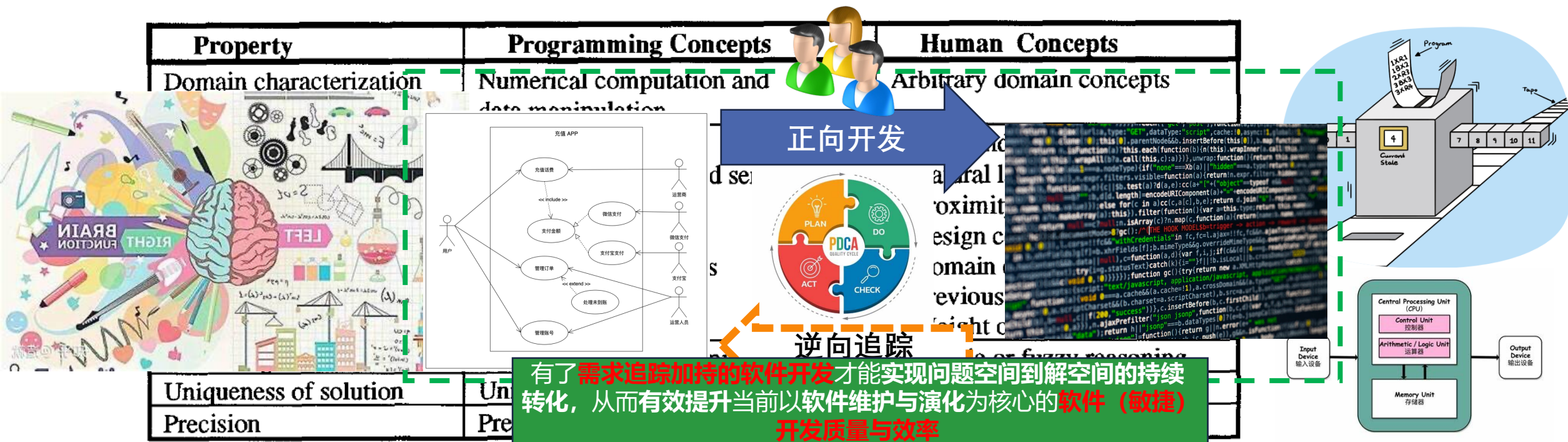


## PART 03

# 基于LLM的需求追踪技术初探

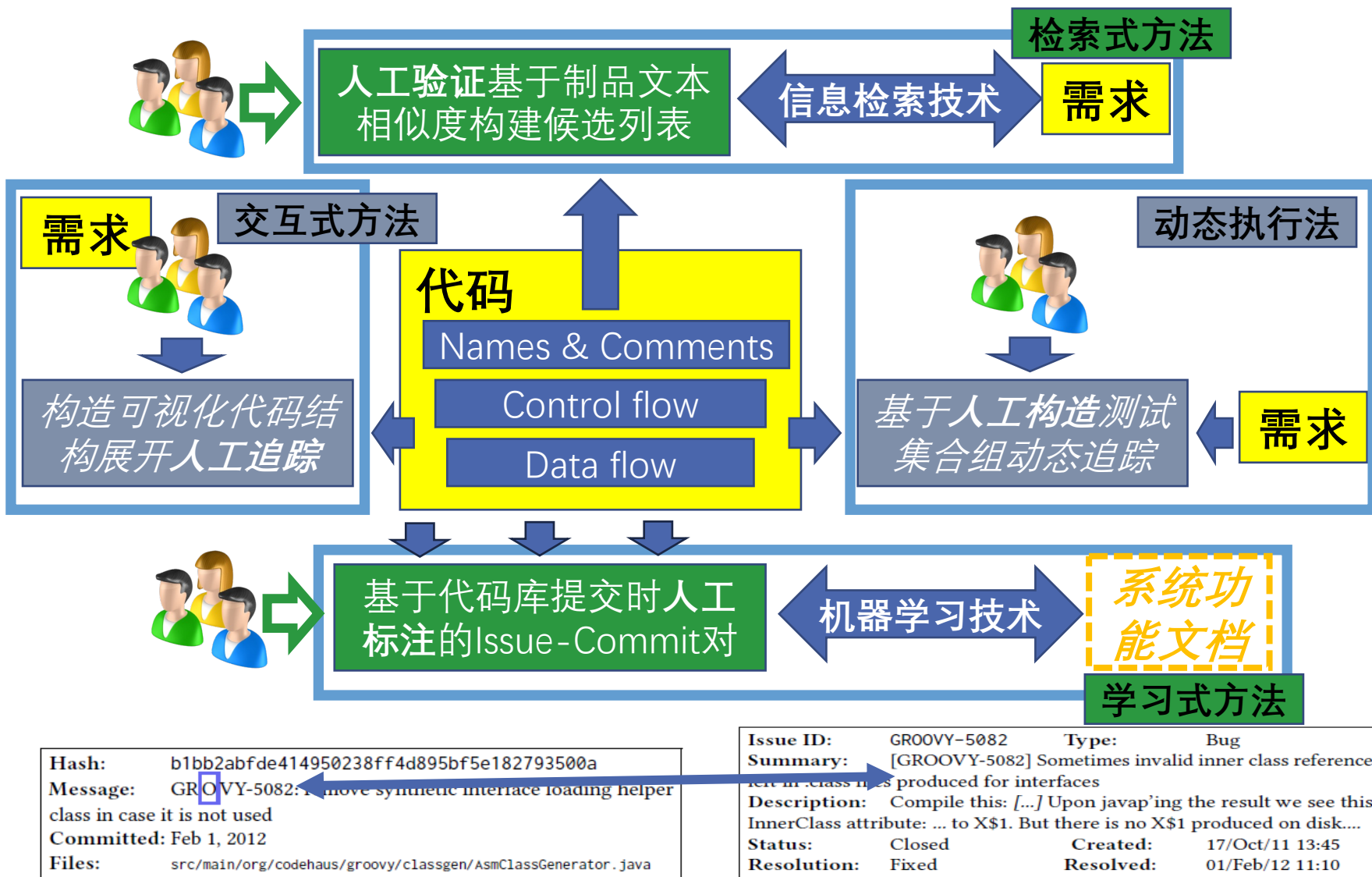
# 需求追踪的本质问题：概念指派 (Concept Assignment)

- 生成或维护需求与代码之间的追踪关系的本质问题，是要将模糊的人脑概念与明确的代码元素关联起来，即所谓的概念指派问题[Biggerstaff et al., ICSE 1993]
- 人脑概念基于自然语言和领域知识，而代码逻辑基于数理逻辑与数学运算，二者存在本质区别
- 需要人的创造跨越二者间的语义鸿沟 - 正向指派即为软件开发，逆向指派即为软件（需求）追踪





# 已有自动追踪技术的本质与趋势



## • (半) 自动追踪 (生成) 技术共通点

- 挖掘用户开发的已有软件制品所包含的系统语义
  - 代码结构、测试集合、制品文本、开发记录等
- 优先降低用户追踪成本, 进而提升追踪质量
  - 辅助追踪 – 构造测试 – 搜索筛选 – 日常标注
- 考虑到当前软件开发与概念指派的本质的本质, 需求追踪无法实现全自动化
  - “Human in the Loop” 式的半自动化



# 基于信息检索的自动追踪生成技术概述

主要基于制品文本增强，以及结合软件制品特性的排序优化。更复杂的IR模型作用有限（因制品文本质量较差）

## 步骤：

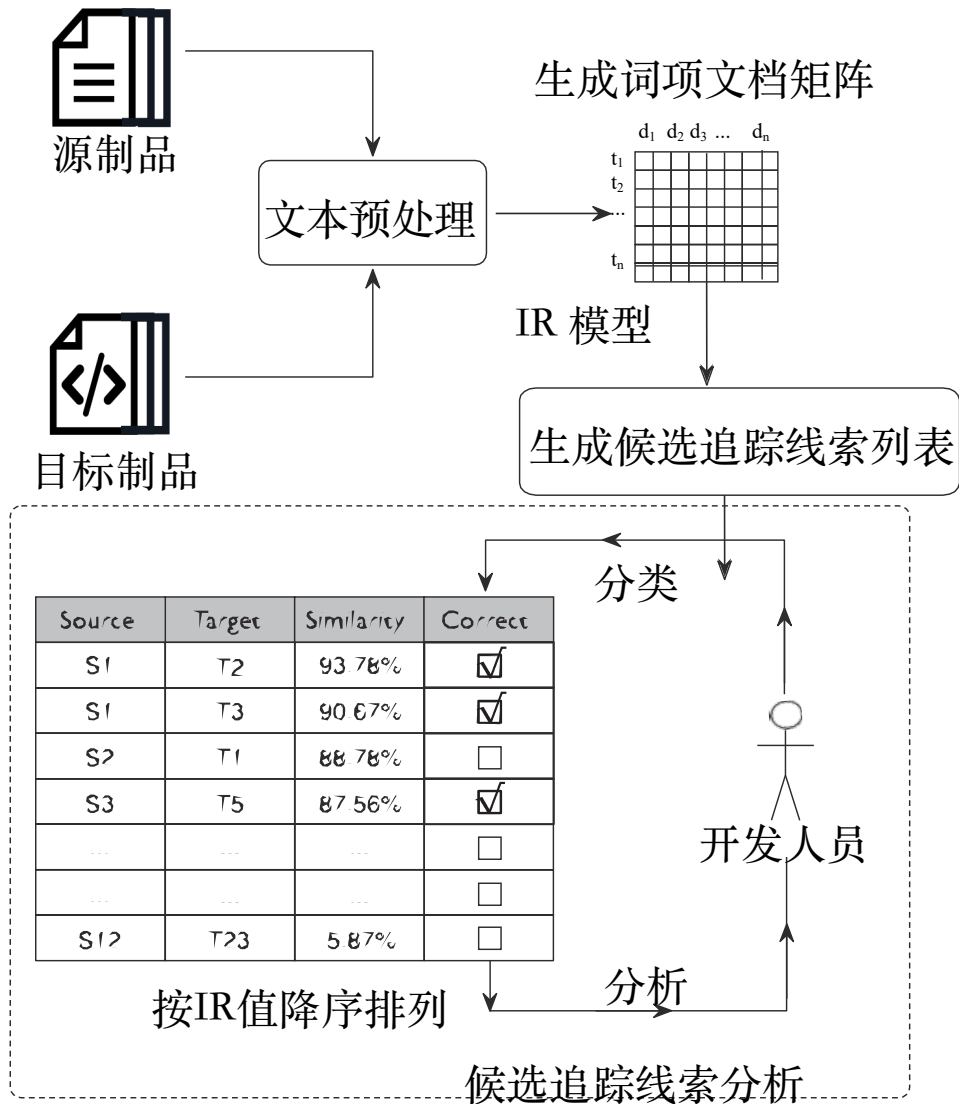
1. 文本预处理
  - 大小写转换、去除特殊字符、去除停用词、提取词干
2. 使用IR模型计算文本相似度，生成候选追踪列表
  - 常用检索模型：VSM, LSI, JSD, BM25, LDA等
3. 分析并优化候选追踪列表，得到最终追踪列表

## 优点：😊

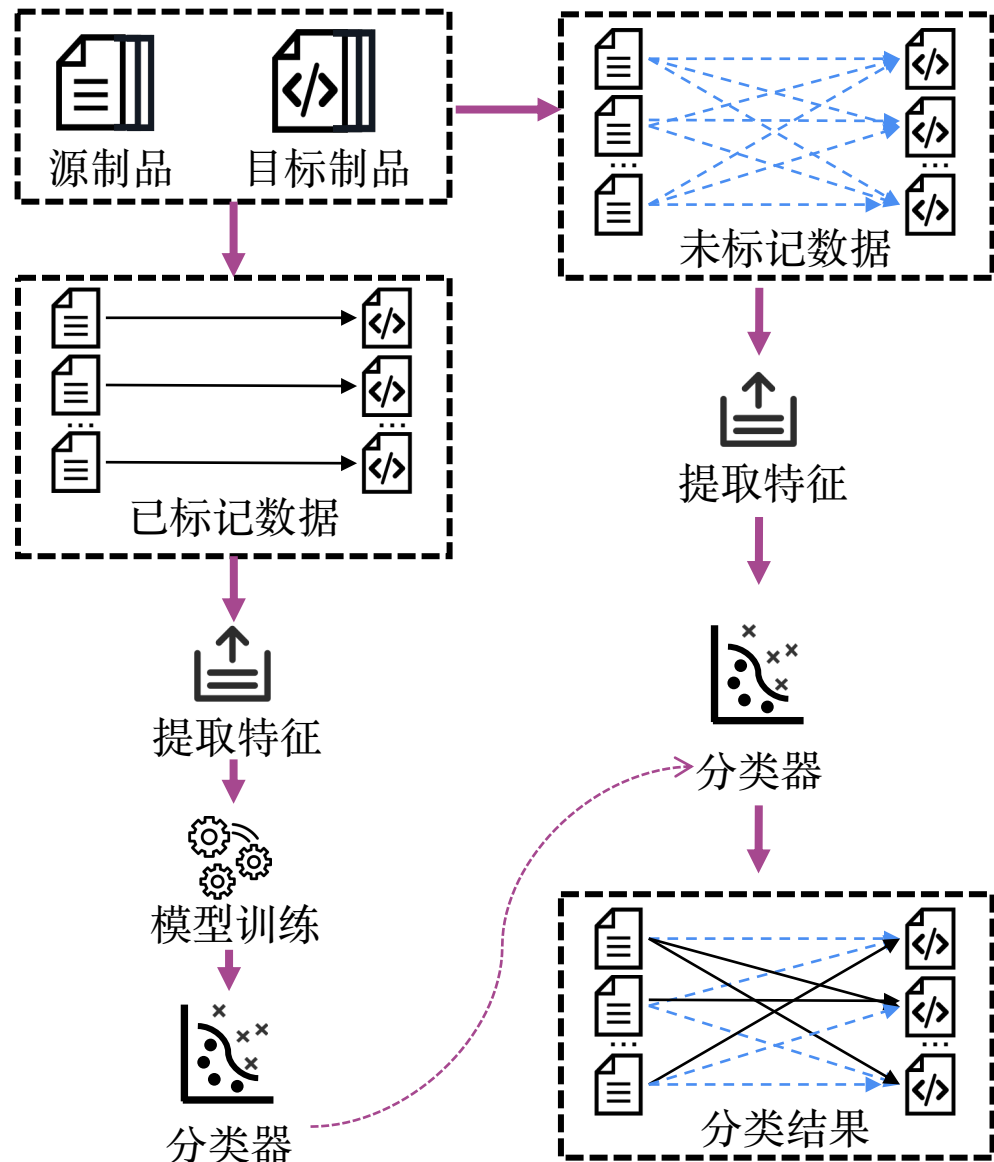
- 方法直观，可解释性强
- 可扩展性强

## 不足：😞

- 语义鸿沟，单词失配
- 对上下文和语义理解能力有限
- 丢失部分非文本特征



# 基于机器学习的自动追踪生成技术概述



相比于经典机器学习应用，可用于学习式自动追踪生成的数据少且质量差，同时面临严重的标签不平衡问题（RTM极为稀疏），需要更全面的特征工程构建，以及多种标签增强技术

## 步骤：

1. 文本预处理
  - 大小写转换、去除特殊字符、去除停用词、提取词干（与检索式方法类似）
2. 特征提取（各类文本相似度是重要的特征组成部分）
3. 模型训练
  - 模型选择、训练设置
4. 使用训练得到的模型预测追踪线索

## 优点：😊

- 模型简洁，可解释性较好
- 可利用先验知识

## 不足：😞

- 特征选择对模型性能影响显著
- 需要高质量的标注数据
- 对复杂问题的解决能力有限

# 基于深度学习的自动追踪生成技术概述

逐步转所有制品向BERT系模型并结合后续增强策略，可为文本构建一个完整模型或为每类制品构建单独模型再联合

词嵌入



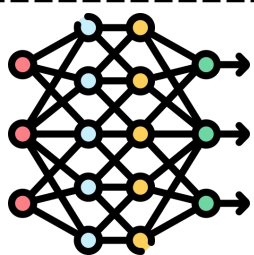
源制品



目标制品

Word2Vec, GloVe, ELMo, BERT...

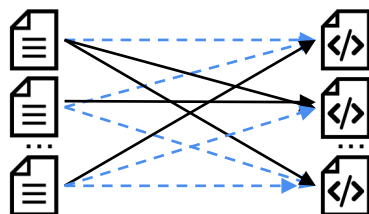
神经网络



RNN, CNN, LSTM,  
GAN, GRU...



追踪线索预测



候选追踪线索

步骤:

1. 基于词嵌入的文本预处理
2. 选择神经网络（含下游模型配置）
  - RNN（LSTM or GRU）、CNN、Transformer
3. 训练神经网络模型
  - 模型构建、训练设置
4. 使用训练得到的模型预测追踪线索

优点: 😊

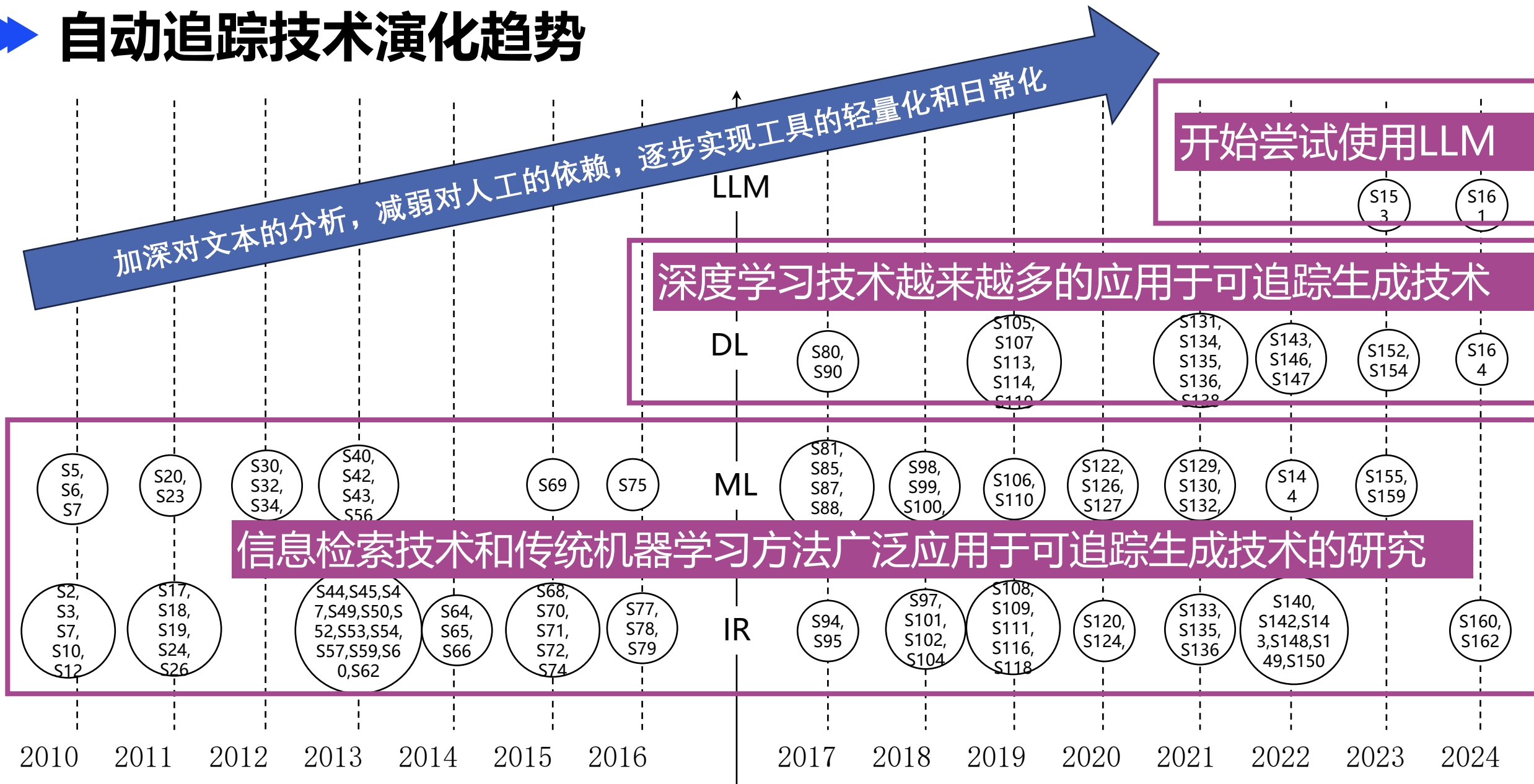
- 减少特征工程需求
- 更好的进行上下文与语义理解

不足: ☹️

- 对计算资源需求较高
- 模型本身可解释较弱
- 需应对可追踪领域数据规模较小



# 自动追踪技术演化趋势



# ▶ 基于大模型的需求自动追踪：进展如何？

## An LLM-based Approach to Recover Traceability Links between Security Requirements and Goal Models

目标模型广泛用于需求工程前期分析

Jameleddine Hassine

jhassine@kfupm.edu.sa

Information and Computer Science Department, KFUPM

Interdisciplinary Research Center for Intelligent Secure Systems, KFUPM

Dhahran, KSA

### 基于GPT3.5的Prompts设计

基于输入的安全目标追踪到了对应的安全需求并给出解释

```
prompt_messages = [
  {"role": "system",
   "content":
     f"You are a requirements engineer. Given the following requirements {Requirements} "
     f"and the following goals {goals}. "
     f"The task consists of linking the security-related requirements within {Requirements} "
     f"to the security-related goals from {goals}. "
     "Note that one goal can be linked to many requirements and vice versa."
     "Provide the rationale for each generated link separately."
     "Do not report non-security related links."},
  {"role": "user",
   "content": ""}]
```

在一个小系统上进行了初步实验，Precision 100%，但漏掉了三个相关trace，重新请大模型判断，大模型又认为均是有关的（随机性）

```
1. **Protect User Privacy**
- **Linked Requirements:**
  - R1: The app shall allow users to set privacy preferences for their designs, controlling who can view or edit them.
  - R2: The app shall provide end-to-end encryption for all user data, ensuring privacy during transmission and storage.
- **Rationale:**
  - Requirement R1 enables users to control the visibility and editing permissions of their designs, thereby enhancing user privacy within the app.
  - Requirement R2 ensures that all user data is encrypted end-to-end, safeguarding user privacy during both transmission and storage processes.

2. **Safeguard Financial Transactions**
- **Linked Requirements:**
  - R3: The app shall integrate with trusted payment gateways like PayPal or Stripe to secure financial transactions.
  - R4: The app shall implement tokenization for sensitive financial information to safeguard against unauthorized access.
- **Rationale:**
  - Requirement R3 ensures that the app integrates with secure payment gateways to safeguard financial transactions, enhancing the overall security of financial data.
  - Requirement R4 implements tokenization for sensitive financial information, adding an extra layer of security to protect against unauthorized access to financial data.
```

# 基于大模型提示工程的需求自动追踪：优点与问题

基于Claude（整体效果更好，  
允许更长的提示输入）

ID	Prompt	Precision	Recall	TP	TN	FP	FN
1	Is (1) related to (2)?	18% 32.4%	46% 92.3%	6 12	225 227	27 25	7 1
2	Ignoring the different abstraction levels of the artifacts, is (1) related to (2)?	17%	54%	7	218	34	6
3	Is there a traceability link between (1) and (2)?	80% 40%	31% 46.2%	4 6	251 243	1 9	9 7
4	Below are artifacts from the same software system, is there a traceability link between (1) and (2)?	12%	100%	13	154	98	0
5	Below are artifacts from the same software system. Ignoring different levels of abstraction, can (2) be decomposed from (1)?	36%	62%	8	238	14	5
6	Below are artifacts from the same software system. Ignoring different levels of abstraction, does (2) fulfill (1)?	67%	46%	6	249	3	7
7	Below are artifacts from the same software system. Ignoring different levels of abstraction, are (1) and (2) part of the same feature and share functionality?	32%	54%	7	237	15	6
8	Below are artifacts from the same software system. Ignoring different levels of abstraction, does (2) implement system functionality from (1)?	22%	77%	10	216	36	3
9	Below are artifacts from the same software system. Ignoring different levels of abstraction, are (1) and (2) dependent on the same system capability?	19%	85%	11	204	48	2
10	Below are artifact from the same software system. Ignoring different levels of abstraction, can (2) be one of the hierarchical decompositions of (1)? Consider whether (2) implements a part of (1).	22%	92%	12	208	44	1
11	Combining all questions and chain-of-thought reasoning.	37.9%	84.6%	11	234	18	2

Insights and Strategies for Prompt Engineering  
Automated Software Traceability

**Human:** What does this artifact mean?  
The DPU-TMALI shall utilize SCM-DCI-SR, along with ERRNO provided by DPU-DCI to decode errors and place them on an error queue for DPU-CCM.

Notre Dame, Indiana

Notre Dame, Indiana

**Assistant:** ... Some guesses at the meaning:

DPU may stand for Data Processing Unit ... SCM-DCI-SR seems to be some kind of protocol or interface used for communication between DPU-TMALI and DPU-DCI.

ERRNO is a standard way for software components to return error codes. So DPU-DCI seems to be providing error codes to DPU-TMALI ... DPU-TMALI is receiving error data from DPU-DCI, decoding or interpreting those errors, and then queuing them up to be sent to DPU-CCM.

DPU-CCM seems to be a component that is receiving error data, perhaps for monitoring purposes...

TABLE III  
MEAN AVERAGE PRECISION OF RANKED TARGET ARTIFACTS

Dataset	Ranking by VSM	Ranking by VSM + LLM
CM1	70.7%	79.4%
iTrust	44.5%	44.2%
Dronology (NL)	82.9%	100%
Dronology (PL)	23.2%	30.8%

初步观察结论：大模型具备一定的制品追踪能力但有限，  
与已有自动追踪技术结合可能是更好选择

## 基于大模型提示工程展开需求追踪的问题

- 对**不同抽象层级制品的语义关联**感到“难以接受”，需要**强调制品来自于同一个系统**
- Prompt提示后有一定效果但不够明显（**结果偏向高查全率**）
  - 提示文本细节**（单词复数、词组顺序）**对结果影响较大**
- 在**词嵌入**方面也**未体现出显著优势**

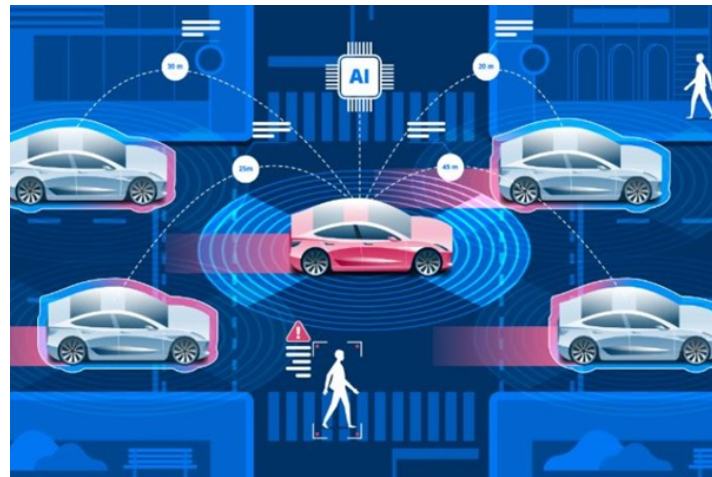


# ► LLM + 需求自动追踪展望：大语言模型的技术特性

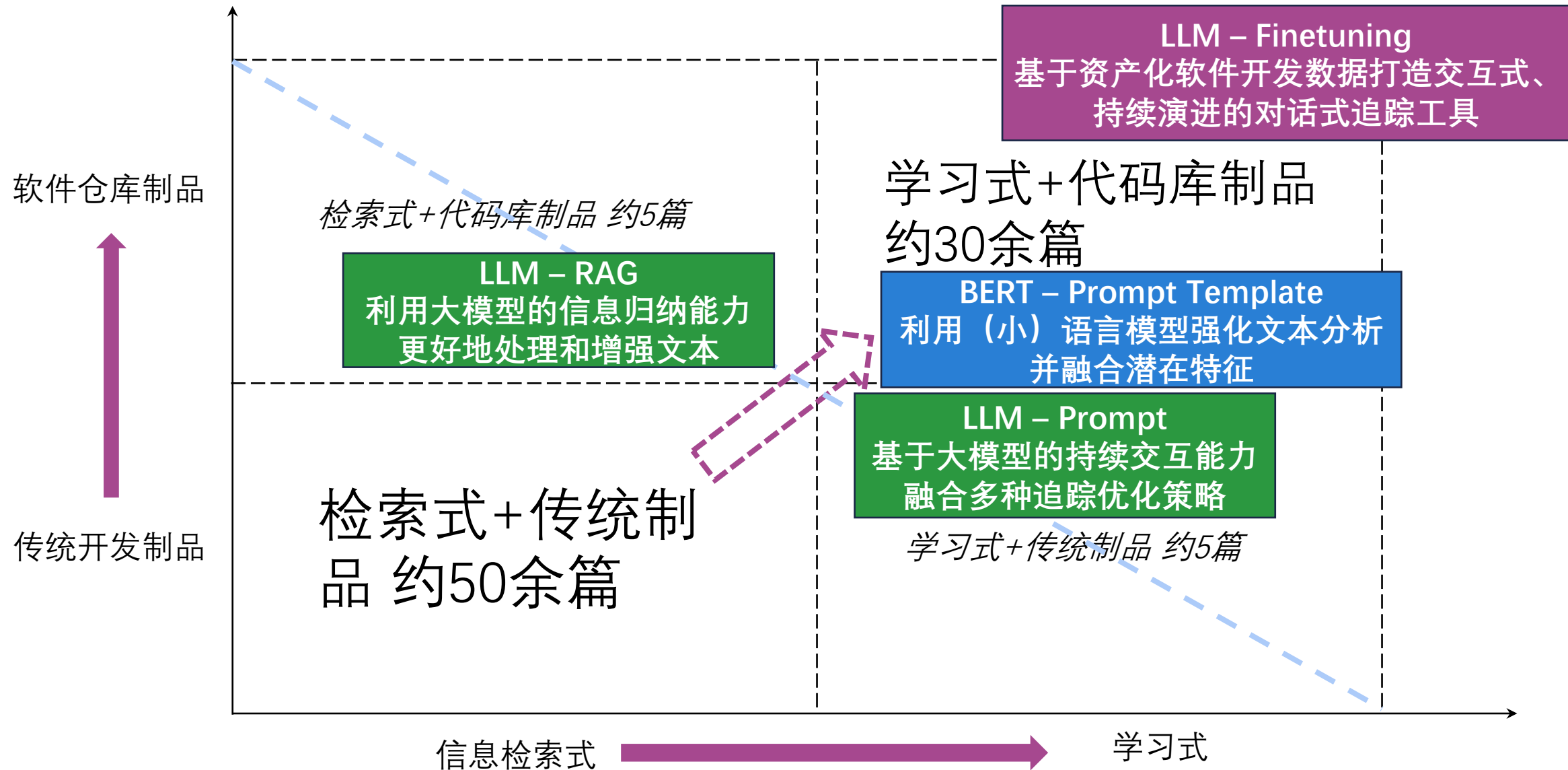
- 大模型的本质独特性 – 计算层面[Rinard@ICSE 2024 Keynote]
  - **大模型的输出**仍是依照用户查询/提示，基于数据训练得到的**概率模型的采样**
  - 这使得用户可以反复提示大模型并从返回的多个结果中挑选出符合需要的
- 大模型的本质独特性 – 产品层面[Chintala@ICSE 2024 Keynote]
  - 可视为一种“**柔性**”**数据库查询**，用户可通过持续交互逐步确定正确答案
  - 被**训练过的任何一块文本片段**都有可能被涉及并**整合到大模型的输出**之中
- 由此带来的问题 – 难以避免的“**幻觉**”问题
  - **大模型的通用性和灵活交互能力本质上基于概率模型所共有的反复多次采样，而这个过程本质是不确定的**
  - **大模型的优质答案依赖于是否在训练中“亲眼见过”相关数据，否则会不可避免的**出现“幻觉”与“编造”****

# ► LLM + 需求自动追踪展望：如何更好地与大模型“共舞”

- 首先接受大模型的“不确定性”
  - 因为这同时是大模型能力的“源泉”
- 从基于控制的系统中学习LLM (AI) 系统设计
  - 融合多种输入源以相互印证
    - 中长期目标：在海量高质量项目数据上训练专用大模型
  - 结构化模型输出以方便快速展开分析
    - 显式引入业务相关的领域模型与奖励模型 – LLM-RAG
  - 人在循环中不断对系统输出进行反馈
    - 引入团队经验与先验知识 – LLM-Prompt
  - 系统工具化时需构建适合的交互界面
    - 便于快速评估大模型的输出结果并及时纠错



# LLM + 需求自动追踪：后续工作展望

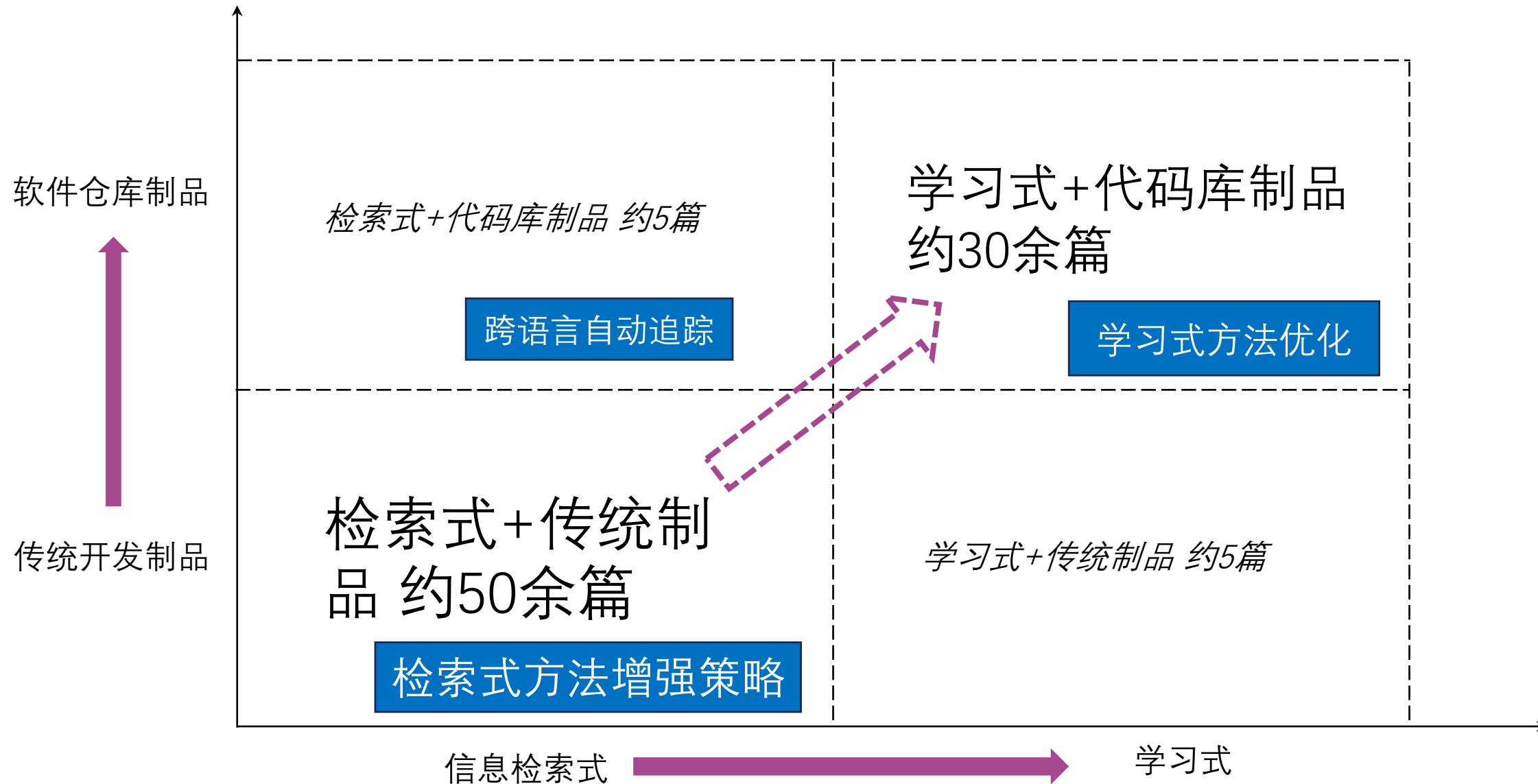




## PART 04

# 本人代表研究简介

## 近十五年来自动追踪研究分布与本人工作



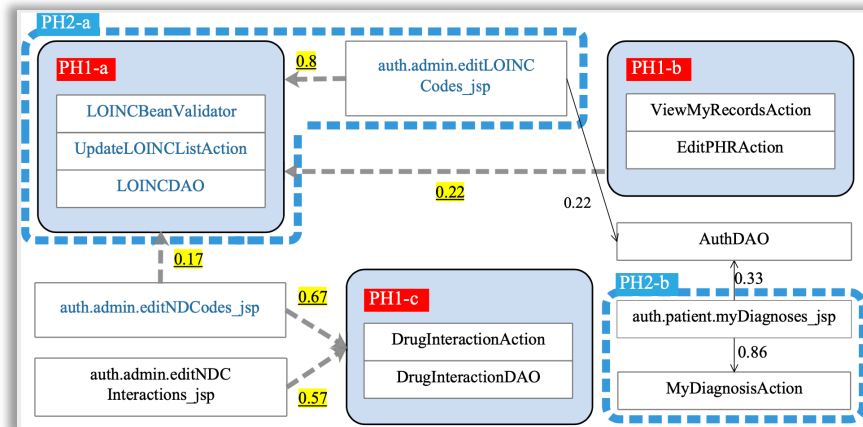
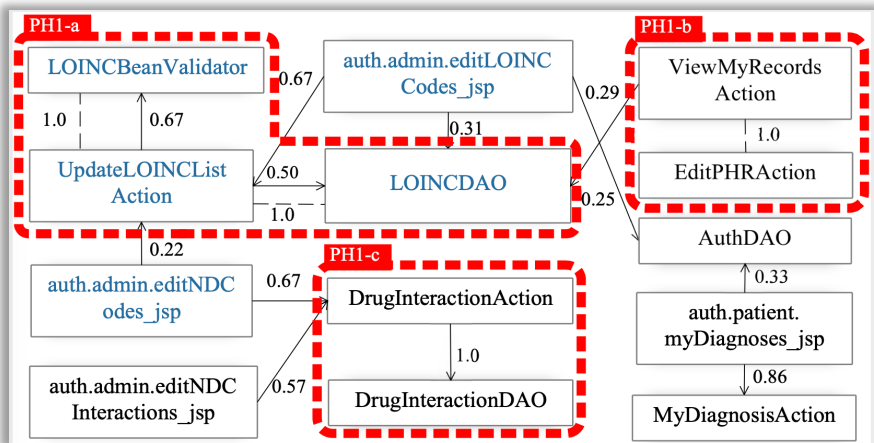
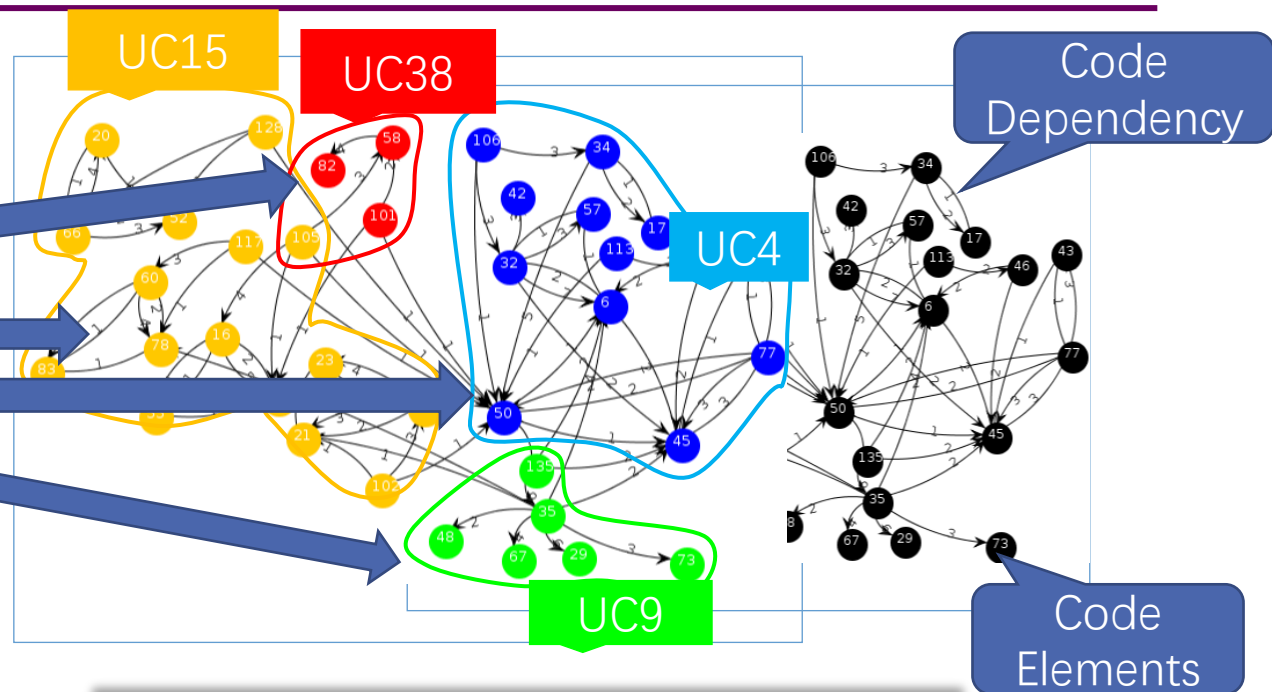
# 基于代码依赖分析的检索式方法增强：CLUSTER' [EMSE 2022]

Class	req	score
<a href="#">editNDCInteractions.jsp</a>	UC15	0.298
<a href="#">DrugCodesBeanValidator</a>	UC15	0.227
<a href="#">DrugCodesDAO</a>	UC15	0.113
<a href="#">editNDCodes.jsp</a>	UC15	0.113
Parameter	UC15	0.112
<a href="#">DrugInteractionAction</a>	UC15	0.054
<a href="#">editDrugCodes.jsp</a>	UC15	0.037
<a href="#">NDCodesDAO</a>	UC15	0.032
<a href="#">AuthDAO</a>	UC15	0.026
<a href="#">UpdateCodesListAction</a>	UC15	0.025
<a href="#">viewResult.jsp</a>	UC15	0.018
<a href="#">UpdateNDCodesListAction</a>	UC15	0.016

User Feedback



仅需少量用户判断（与代码依赖分析结合）即可有效改善检索式方法精度

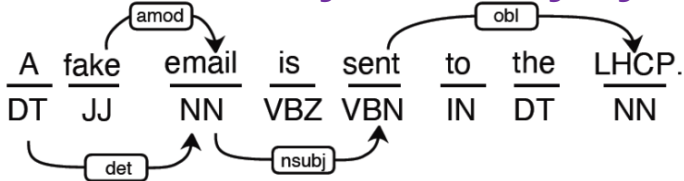






# 基于共识词对的检索式方法增强：TAROT<sup>[ASE 2022]</sup> & TRIAD<sup>[ICSE 2024]</sup>

## TraceAbility Recovery by cOsensual biTerms (TAROT)



### UC35 Report Adverse Event Use Case

#### 35.1 Preconditions:

The iTrust user (patient) has been authenticated in the iTrust Medical Records system (UC3).

#### 35.2 Main Flow:

A patient selects to report an event related to a prescription drug [S1] or immunization [S2] reaction.

#### 35.3 Sub-flows:

[S1] A fake email is sent to the LHCP who prescribed the medication indicating the patient name and MID, drug, and symptoms.

[S2] ...

#### 35.4 Alternative Flows:

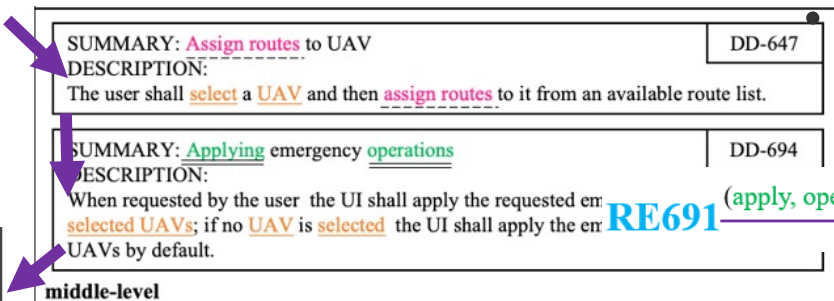
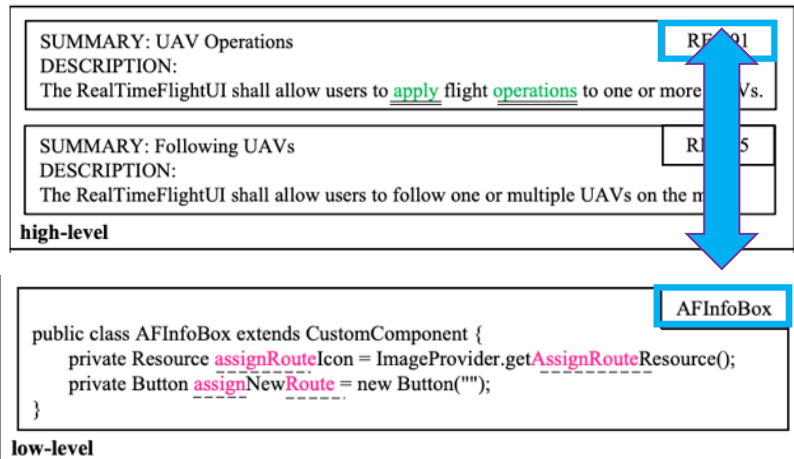
None

```
2 * Sends email to users. Since we don't want to train spammers in
3 * 326, this just inserts into a database.
4 ...
5 */
6 public class EmailUtil {
7     private DAOFactory factory;
8     public EmailUtil ( DAOFactory factory ) {
9         this.factory = factory;
10    }
11    public void sendEmail ( Email email ) throws DBException {
12        factory.getFakeEmailDAO().sendEmailRecord ( email );
13    }
14 }
```

- 参照社交文本主题建模中的词对增强策略，引入软件追踪 共识词对 探索跨越制品开发的共识语义

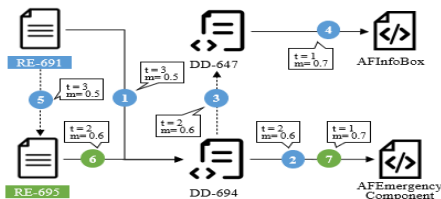
- 使用词性分析和代码标识符分析来抽取共识词对，例如图中的(send, email) 和 (fake, email)
- 利用共识词对扩充制品文本并构建检索优化策略，在9个系统上帮助提升4.8%至21.8%MAP

## Traceability Recovery by blterm-enhAnced Deduction of transitive links (TRIAD)



- 充分利用已有制品与其文本，在五个实验系统上平均优于四个基线方法10%MAP

## 基于TAROT的共识词对，在多制品中推导语义传递关系以优化检索



# ► 基于共识词对的跨语言（中英）需求追踪：AVIATE<sup>[ASE 2024]</sup>

- 软件开发全球化导致软件制品文本出现多语言混用现象，阻碍自动追踪

- GitHub上12.7%的commit logs由非英文书写，最常用为中文28.6%，其余依次为西葡法日等
  - 国内公司实践中更常见（代码仍以英文为主）
- 可通过翻译统一语言，但会出现多种翻译版本（翻译变体），削弱自动追踪技术效果
- 在多翻译变体上抽取共识词对增强文本并设计优化策略，在17个跨语言系统上AP提升31.4%，MAP提升11.2%
  - 英汉项目14个，来自阿里、腾讯、网易、小米、百度等公司开源项目
  - 英韩、英日、英德项目各一（提升高于英汉）

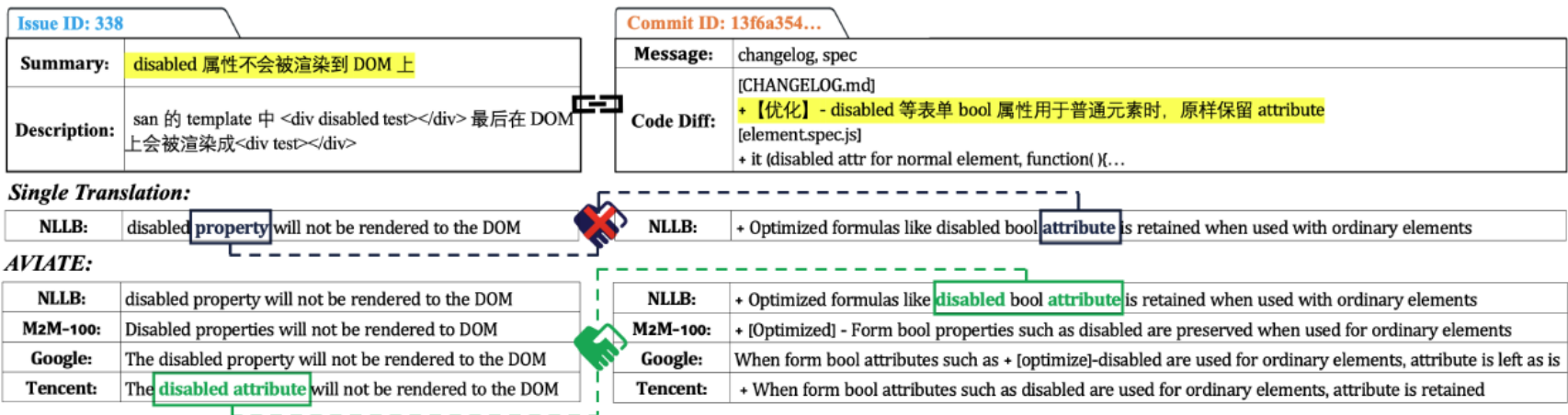


Figure 1: Motivating example adapted from the San project.

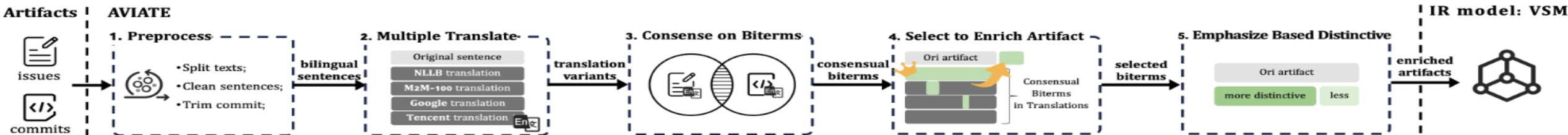


Figure 2: Overview of the AVIATE framework.

# ► 基于增强预处理的學習式方法: SPLINT [FSE 2022]

## Semi-Supervised Pre-processing for LearnIng-based Traceability (SPLINT)

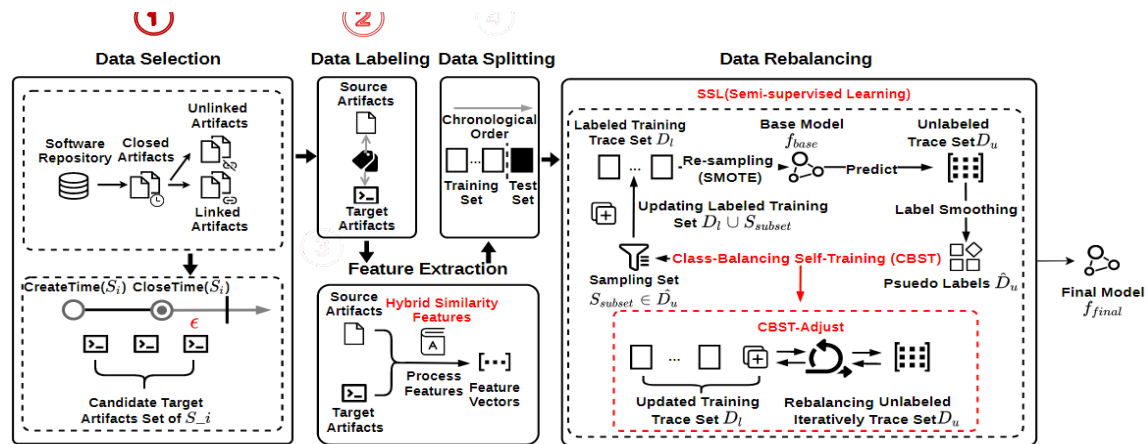
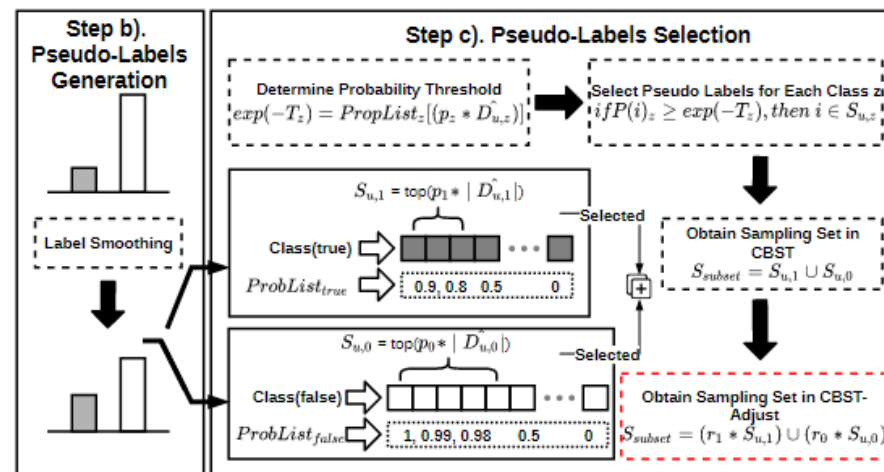
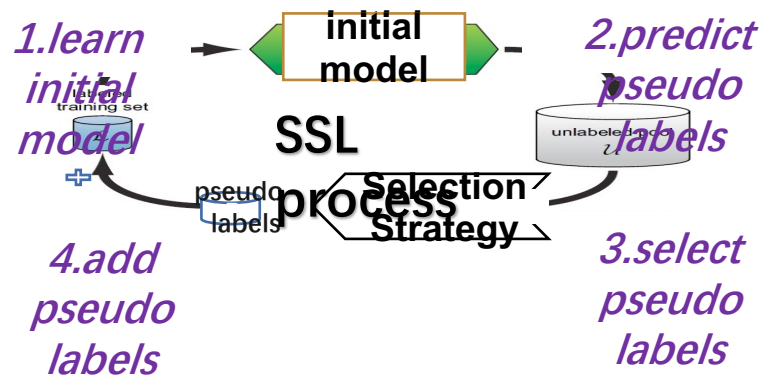


Figure 3: Overview process of SPLINT

- 基于已知的Issue-Commit对进行自动化追踪
  - 提交Commit时往往会标记所解决的Issue
  - 标签严重不平衡，数据稀疏，影响分类效果
- 基于类别平衡和自学习技术预处理数据的分类优化
  - 6个开源系统上平均提升0.76%AUC
  - 10个工业系统上平均提升8.39%AUC



Class-balancing and self-training(CBST) selection process



# 探索中工作：BERT在基于制品文本的自动追踪潜力

## 目标：

1. 对学习式追踪的Issue-Commit对进行语义聚合
2. 基于历史Issue的Component标签构建分类模型

## 思路：

1. 考虑软件开发过程特性（**人员协作&逐步演化**）
2. 强化模型语义能力
  - 使用BERT预训练语言模型
  - 使用提示学习克服语义鸿沟
3. 设计一个能够增量分类issue新标签的推荐方法

## 效果：

1. 提升任务经典指标Recall@1 3.2% - 215.7%
2. 消融实验说明方法各部分均对效果起正向作用

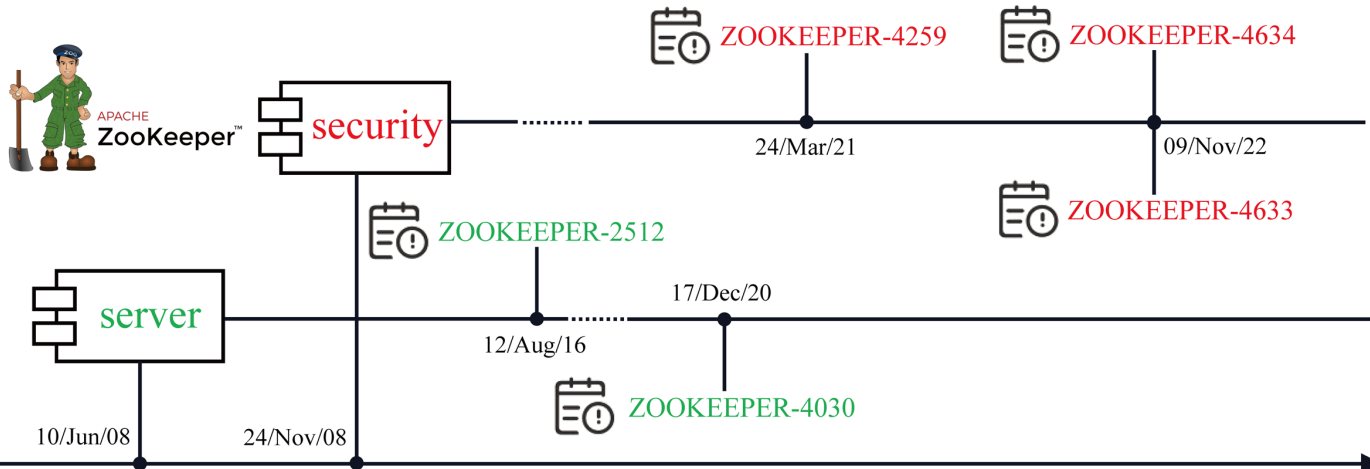
## 经验分享：

1. BERT类“小”语言模型比LLM更容易拟合给定数据
2. BERT提示模板可有效引入额外知识，实践中可**选**择通过LLM补充重要上下文，再利用BERT拟合

引入基于BERT的提示学习，并在提示模板中融合软件开发过程特性

**<text>**. This issue is created by **<creator>** in **<created time>**. This is **<component>** component.

Issue ID	Summary	Creator	Created time
ZOOKEEPER-2512	Allow Jetty dependency to use HTTPS and Basic Auth	Edward Ribeiro	12/Aug/16
ZOOKEEPER-4030	Optionally canonicalize host names in quorum SASL authentication	Damien Diederer	17/Dec/20
ZOOKEEPER-4259	Allow AdminServer to force https	Norbert Kalmar	24/Mar/21
ZOOKEEPER-4633	Provide digest auth support for Admin Server APIs	Li Wang	09/Nov/22
ZOOKEEPER-4634	Provide x509 auth support for Admin Server APIs	Li Wang	09/Nov/22



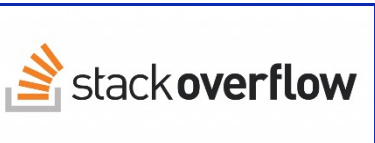
# 科研情况简介

右侧为论文和项目图例，  
加粗论文为一作/通讯 →

- CCF-A
- CCF-B
- CCF-C
- 即将/已投稿
- 企业技术合作



GitHub 2.84亿+  
公开软件仓库



Stack Overflow  
5800万+ 提问&回答

## 面向软件协作网络的功能追踪与开发管理

目标：显著提升日常软件开发的效率与质量，有效支持数智化时代BizDevOps实践

DevOps国家标准起草  
金融科技、智能电网合作  
vivo软件产品工作坊设立  
数智化转型课程共建

### TRACEABILITY



## 系统功能追踪

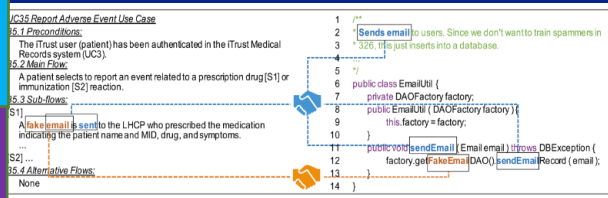
核心 | 自动化软件可追踪

- ICSM2012
- JSEP2015
- SANER2017
- ICPC2019
- EMSE2022
- ASE2022
- ICSE2024
- ASE2024

检索式追踪线索  
生成与优化

版本注释与多版本  
代码diff自动追踪

开源防投毒社区可疑活动分析



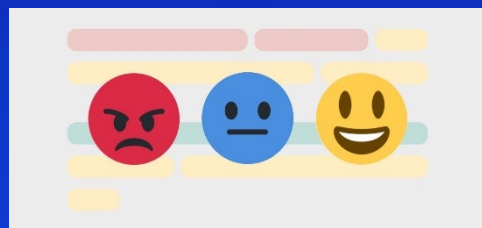
学习式追踪线索  
生成与维护

基于功能关联的  
Issue分类与聚合

FSE2022

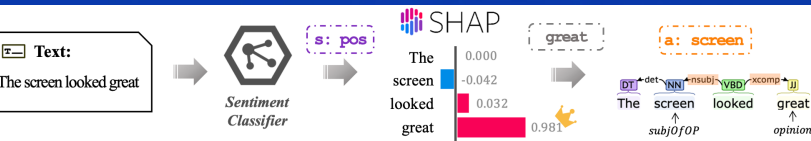
FSE2025拟投

ICSE2025已投



## 开发状态管理

核心 | 情绪分析与软件开发



ICPC2021

ICSE2025已投

基于软件工程  
文本独特性的  
情绪分析优化

基于情绪分析的软  
件开发状态感知

研发外包产出评估技术

Internetwork2022

ECAI2024已投

基于深度学习和可  
解释框架的细粒度  
情绪诱因分析

开发协作网络分析  
与代码评审者推荐

ICSE2022

COMPSAC2022

# PART 05

## 总结与展望



## ► 个人观点总结 (for “Take away” )

- 需求可追踪性指在软件系统中将各类制品基于系统功能建立并维护显式语义关联
  - **普通开发者**快速融入团队，有效提升**开发效率与质量**的重要途径（容易忽略）
  - **项目管理者**治理开发过程，实现**开发数据资产化**的必经之路
- 以需求追踪为核心的开发数据资产化是大模型赋能软件开发的关键
  - **需求可追踪性能够在细粒度的源码层面支持大量软件开发相关任务**
  - **面向需求追踪实现开发数据资产化是在软件开发中充分利用LLM能力的重要前提**
- 需求自动追踪长期以来是软件工程领域研究与实践的热点与难点
  - 当前**主流追踪技术**为面向**传统制品文本**的信息检索和面向**代码库文本**的机器学习
  - 在高质量数据有限的前提下可以重点关注**大模型RAG和Prompt与传统分析技术的结合**



# THANKS

