武汉大学 人工智能与软件工程暑期学校 2020-08

演讲人：李必信

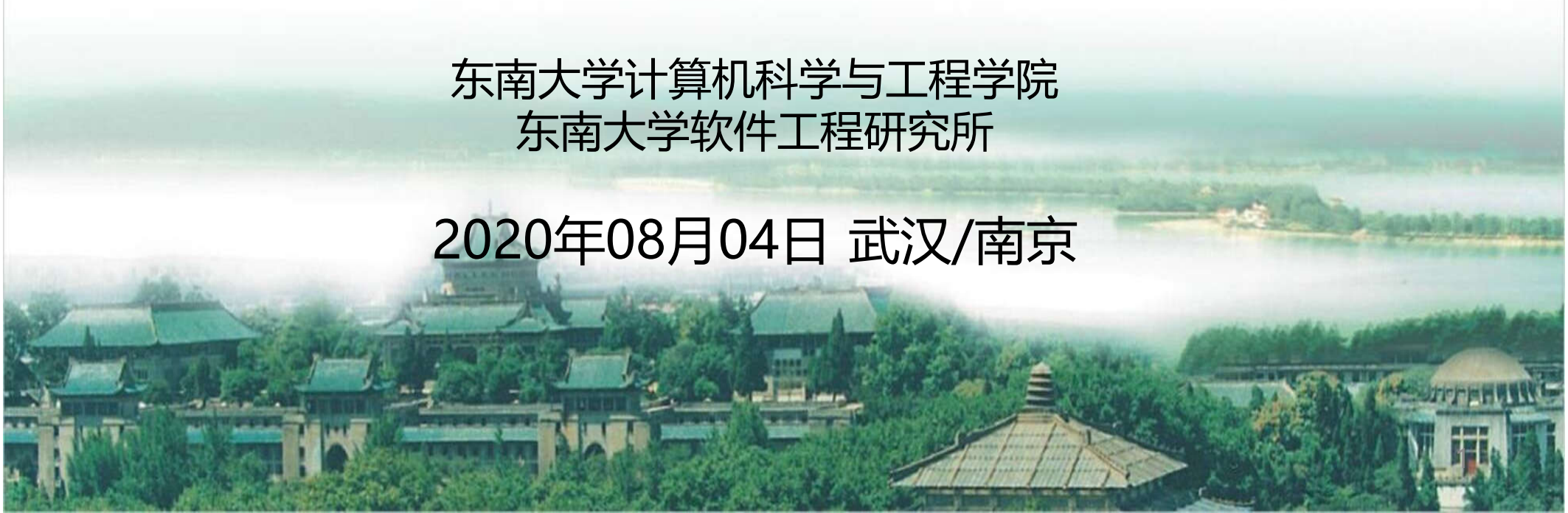
1969年8月出生，2001获得南京大学博士学位，2001-2003分别在芬兰、挪威和荷兰等地从事博士后研究工作。东南大学计算机科学与工程学院教授（二级）、博士生导师，东南大学软件工程研究所所长。2006年入选教育部新世纪人才培养计划、2011年度获得中创软件人才奖、2014年获得教育部科学技术奖（自然科学奖）二等奖（排名第一）。江苏省计算机学会软件专委副主任，第二届江苏省软件工程标准化技术委员会主任委员，中国计算机学会软件工程专委会委员、容错计算机专委会常务委员。研究方向为软件建模、分析、测试与验证、智能软件架构理论和方法、软件演化和软件质量保证等。他从1999年开始，主持各类基金和企业合作项目30多项，先后在《IEEE Transactions on Network and Service Management》、《ACM Computing Surveys》、《Information and Software Technology》、《Software Testing, Verification and Reliability》、《Journal of Systems and Software》、《Science China: Information Science》、《Journal of Computer Science and Technology》、《软件学报》、《计算机学报》、《计算机研究与发展》国内外著名期刊和ICSE、FSE、ASE等重要国际会议发表学术论文180余篇，出版专著3部，教材1部，授权发明专利40多项。相关论文被SCI/EI检索200多篇次，被他人引用累计超过5000篇次。

软件架构智能化：挑战和机会

李必信 博士、教授、博导

东南大学计算机科学与工程学院
东南大学软件工程研究所

2020年08月04日 武汉/南京



目录

一、什么是软件架构?

- A critical issue in the design and construction of any complex software system is its architecture: that is, *its gross organization as a collection of interacting components*. A good architecture can help ensure that a system will satisfy key requirements in such areas as **performance, reliability, portability, scalability, and interoperability**. A bad architecture can be disastrous. [David Garlan 2000]
- Software architecture typically plays a key role as a bridge between requirements and implementation

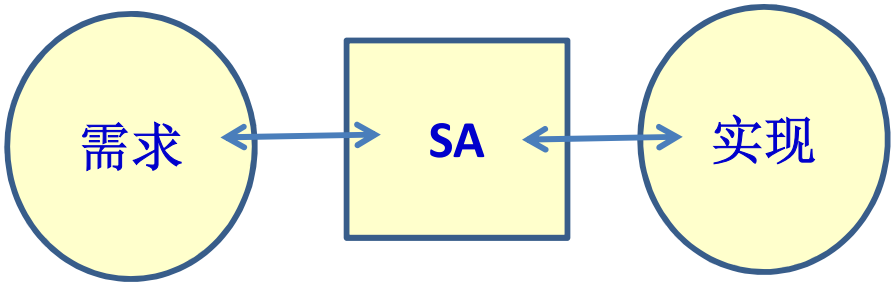


Figure 1：Software Architecture as a Bridge

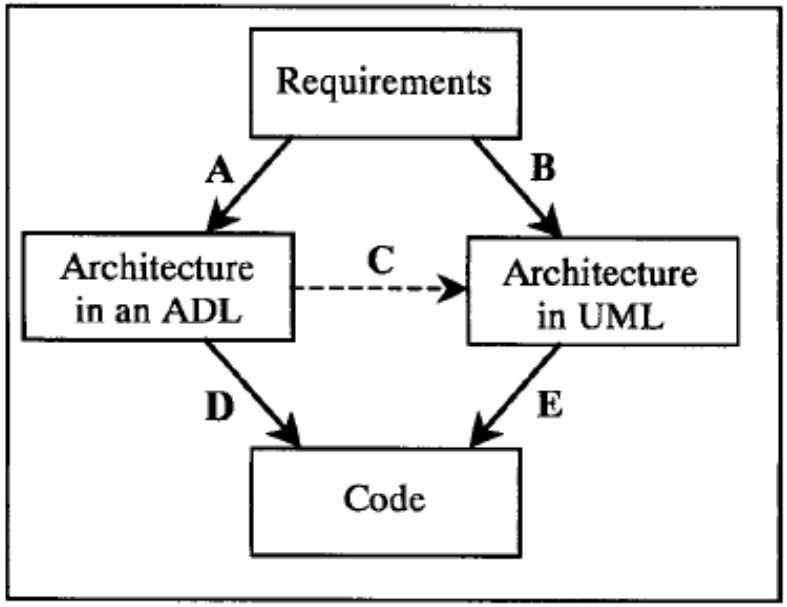


Figure 2: Software Architecture as a Bridge

Path A-D is one in which an ADL is used as the modeling language. Path B-E is one in which UML is used as the modeling notation. Path A-C-E, is one in which an architecture is first represented in an ADL, but then transformed into UML before producing an implementation.

二、什么是智能化软件架构?

- 智能化软件架构(Intelligent Software Architecture, ISA)，是指人工智能(AI)和传统软件架构(SA)结合产生的一种新型软件架构,是一种具有智能的软件架构。
- AI和SA结合主要有两种类型： AI for SA和SA for AI-based System

人工智能与软件架构

焦烈焱 EAII-企业架构创新研究院 常务理事

https://blog.csdn.net/weixin_45443931/article/details/98869387

2.1 AI for SA:

智能化的软件架构（SA）构建过程，而SA自身不一定具有智能。

- AI技术用来辅助SA的构建
- AI for SE

2.2 SA for AI-based System

智能系统的SA，SA自身具有智能，即SA应该至少具备如下能力：感知能力、学习能力、协同能力、推理能力、决策能力，等等

- 机器人控制系统的架构：传感器、环境感知、智能决策…
- 自动驾驶系统的架构：传感器、环境感知、智能决策…
- 自动停车系统：传感器、环境感知、智能决策…
- 智能家居：传感器、环境感知、智能决策…
- 智慧医疗：机器学习、推理、智能决策…
- ……

Special Issue Call for Papers:

- **Journal of Systems and Software**
- **2020 Special Issue Call for Papers: Software Architecture and Artificial Intelligence**
- **https://www.journals.elsevier.com/journal-of-systems-and-software/call-for-papers/software-architecture-and-artificial-intelligence**

Important dates

- Submission Deadline: 9 November 2020
- Initial Author Notifications: 8 March 2021
- Initial Author Revisions Due: 10 May 2021
- Author Notifications for First Revision: 5 July 2021
- Final Author Revisions Due: 6 September 2021
- Final Author Notifications: 11 October 2021

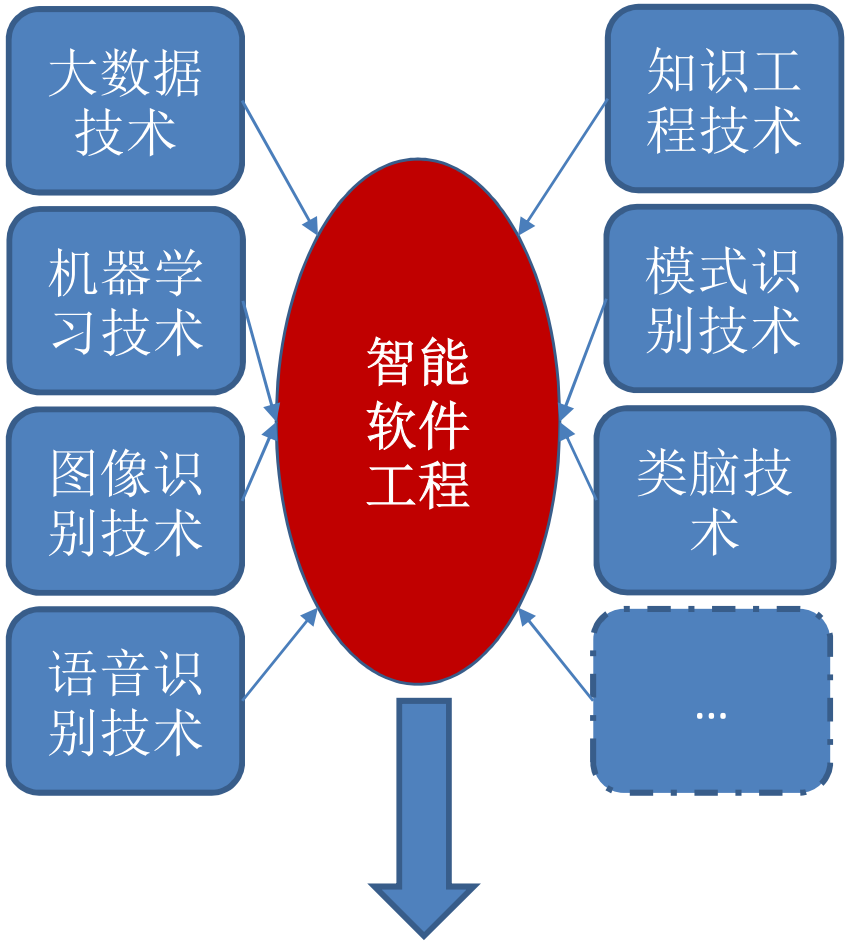
- **Topics of interest include, but are not limited to:**
- Quality attribute concerns for **AI-based systems**
- Patterns and tactics for AI-based systems
- Experiences designing and deploying AI-based systems
- Analysis techniques for uncovering architecture issues in AI-based systems
- Verifying and validating AI models as part of system architectures
- Deployment and maintenance of AI-based architectures
- Method and techniques for improving the architecting process of AI-based systems
- Data and related challenges in architecting AI-based systems
- Monitoring and sustaining AI-based systems architectures
- Iterative and incremental architecting of AI-based systems
- Impact of infrastructure concerns in architecting AI-based systems
- The impact of different algorithms, AI approaches and architecture challenges
- Data for AI research towards improving architecture methods and techniques
- Use of AI to improve architecture, design, conformance and quality
- Architecting data/ML pipelines

2.1 AI for SA

AI for SA一般是指智能化的软件架构（SA）构建过程。

- 方法1：在软件架构构建的每个步骤中充分使用AI技术，帮助获得更好的架构需求、架构设计、架构文档、架构评估、架构实现以及架构演化和维护等。
- 方法2：利用机器学习、知识库、模式库等等进行自动化的软件初始架构生成、评估、演化和优化等。

AI时代的软件工程技术



智能化软件架构

- 是否提高软件能力？
  - 更多应用系统架构
- 是否提高软件质量？
  - 稳定性
  - 可靠性
  - 安全性
  - 可靠性
  - 可信性
  - 扩展性
- 是否提高软件系统性能？
  - 运行时间
  - 资源占用率
- 是否降低开发成本和运维成本？
  - 时间成本
  - 资源成本
  - 经济成本

哪些没有发生变化？哪些发生了变化？

方法1：在SA软件架构构建过程中使用AI技术

●在软件架构构建的每个步骤中充分使用AI技术，帮助获得更好的架构需求、架构设计、架构文档、架构评估、架构实现以及架构演化和维护等。

●软件架构构建过程如图3所示：

（1）AI+SA需求获取；（2）AI+SA设计和建模；（3）AI+SA文档化；（4）AI+SA评估；（5）AI+SA实现和测试；（6）AI+SA演化和维护

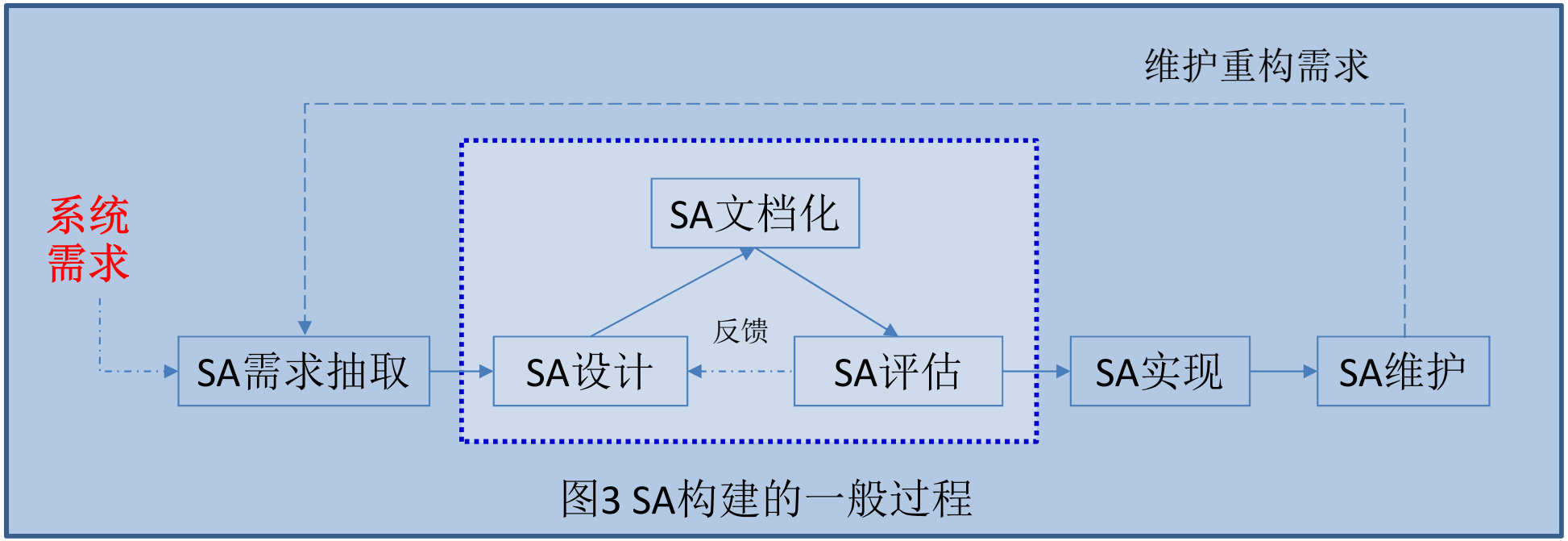


图3 SA构建的一般过程

① AI+SA需求获取

■输入

- 用户需求
- 技术环境
- 架构师经验

■输出

- 需求场景列表：部署哪些功能？主要应用场景：含成功和失效场景
- 架构质量列表：哪些是用户关注的质量属性？哪些是特别关注的？例如：安全性、可靠性、性能、易维护
- 利益相关者列表：投资者、用户、系统管理员、开发团队...

■参与人员

- 系统工程师、网络工程师、软件架构师

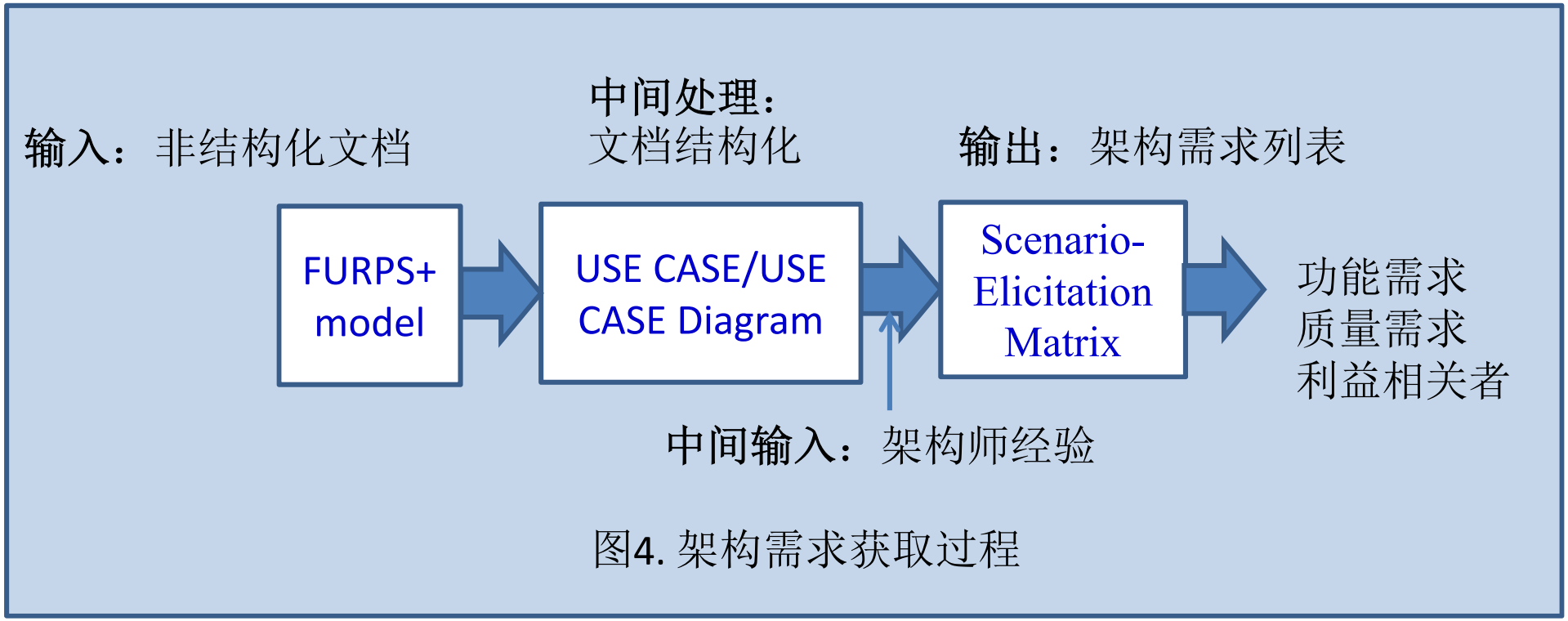


图4. 架构需求获取过程

家庭住宅
酒店宾馆
教室
政府办公楼
幼儿园

使用场景不同，需求不同，结构就不同！

功能需求
质量需求
利益相关者

图5. 结构和功能的关系

There are three outputs from the process: an enumeration of functional requirements made concrete by use cases, an enumeration of specific architectural requirements, and an enumeration of a collection of quality scenarios that provide concrete tests for the architectural requirements.

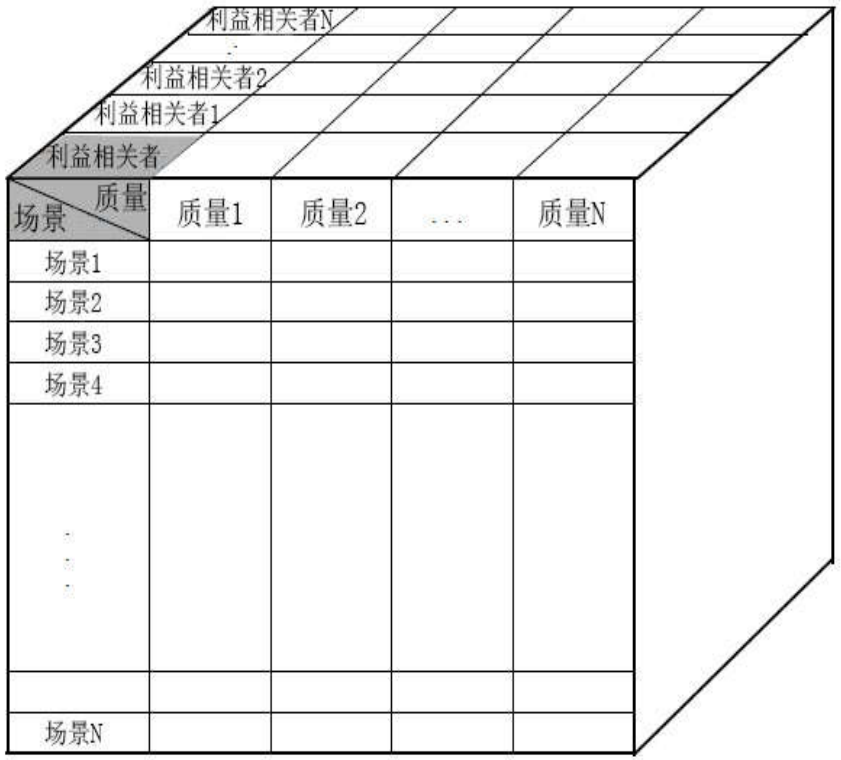


图6. A Scenario-Elicitation Matrix

■Quality Scenarios

- **Abstract Scenarios** is the *generic* requirement of a family of systems.
- **Quality-Specific Scenarios** for making the quality requirements concrete.

如何运用AI技术？

- 自然语言理解NLP、NLU
- 深度学习CNN
- 强化学习
- 知识库
- 知识网络
- 知识本体
- 知识图谱
- 大数据
- 语音识别
- 图像识别
- ……

例如：需求获取过程如图7所示

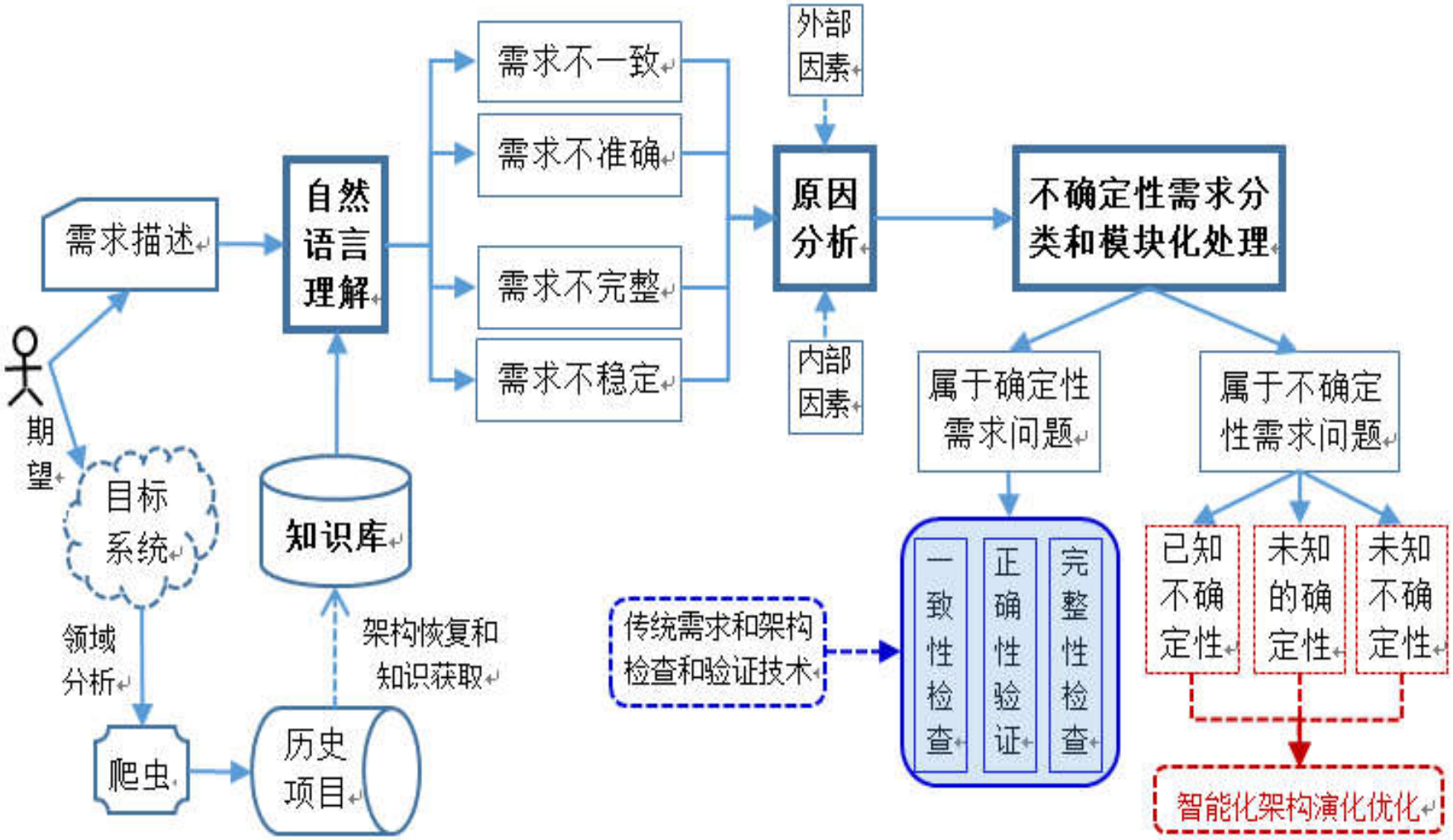


图7. 一个需求获取和验证过程

② AI+SA设计

- 输入：
  - 质量场景
  - 架构风格
  - 设计模式
  - 设计原则
  - 架构需求
- 输出
  - 架构设计文档
- 参与人员
  - 软件架构师
  - 软件工程师

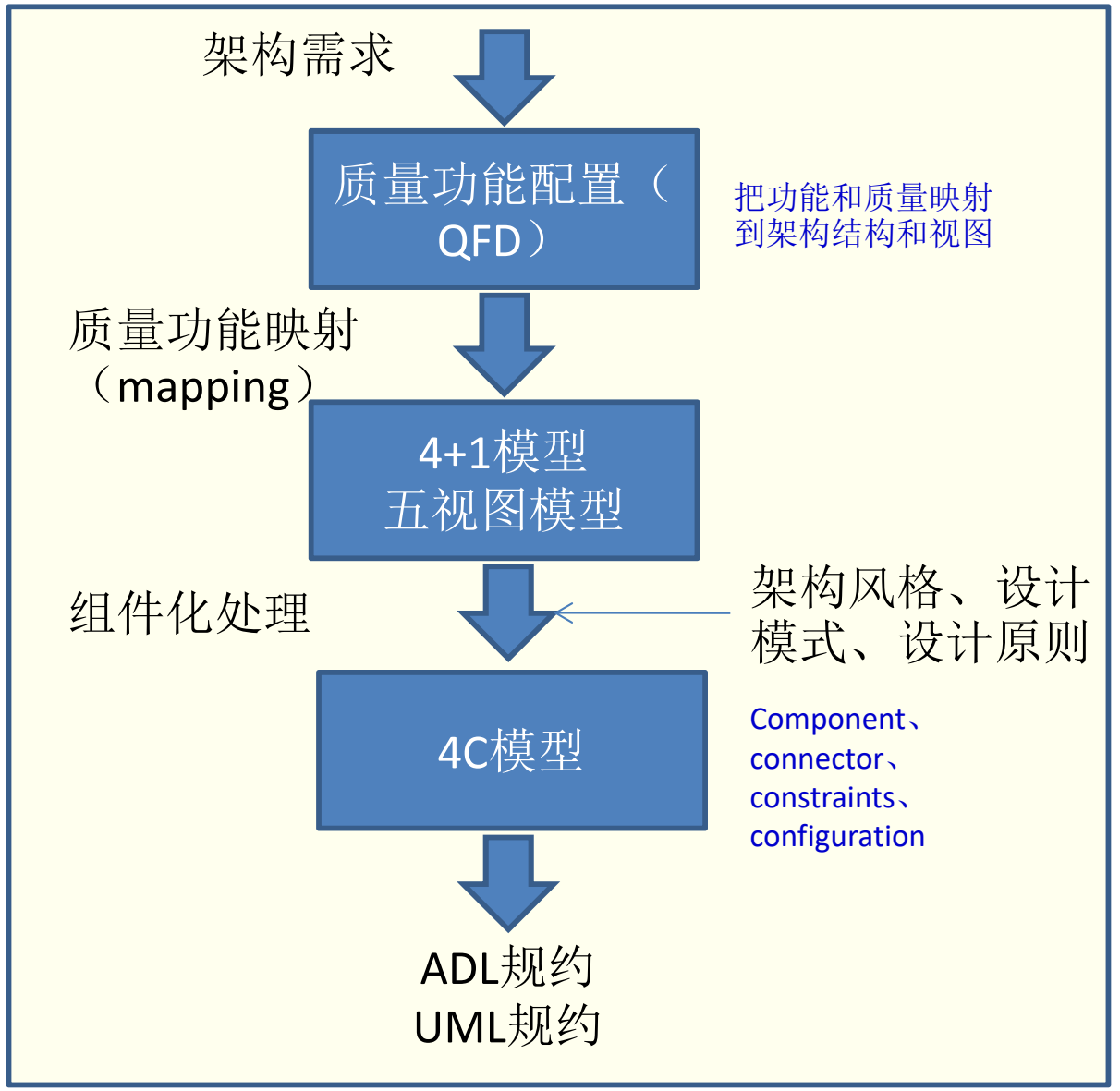


图8. 架构设计建模过程

- **Architectural Structures and Views**
  - Functional structure
  - Code structure
  - Concurrency structure
  - Physical structure
  - Developmental structure

Table 1: Views Useful for Different Qualities

Quality	Useful Structures
Performance	Concurrency, Physical
Security	Concurrency, Code
Reliability/ Availability	Concurrency, Physical
Modifiability/Maintainability	Functional, Code, Development

③ AI+SA文档化

- An architecture's documentation is designed to support the needs of the programmers and analysts. It can be a tremendous vehicle for enhancing communication among the stakeholders and for eliciting architectural requirements from them.
- Creating and maintaining the architectural documentation, as graphically depicted in Figure 9, is a critical success factor in a long-lived software architecture.

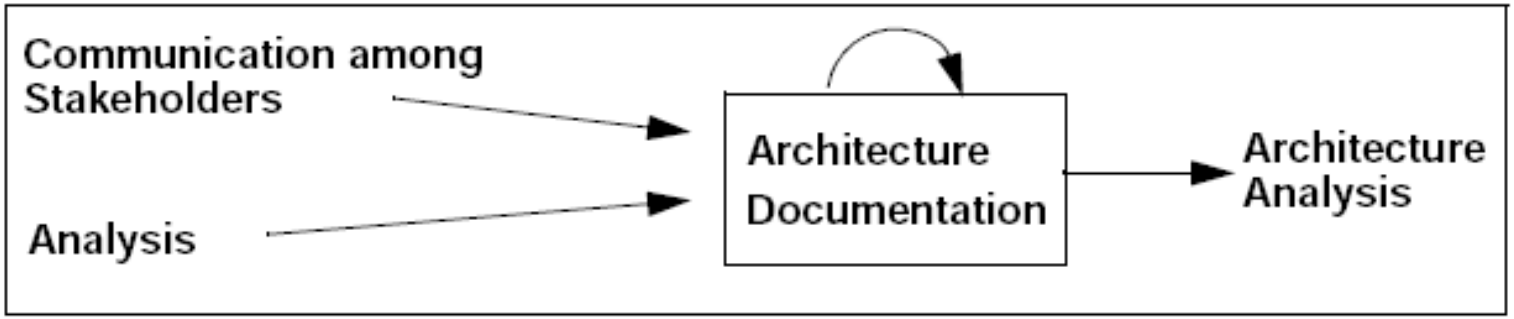


图9. 架构文档化

案例1： Software Architecture Documentation: The Road Ahead

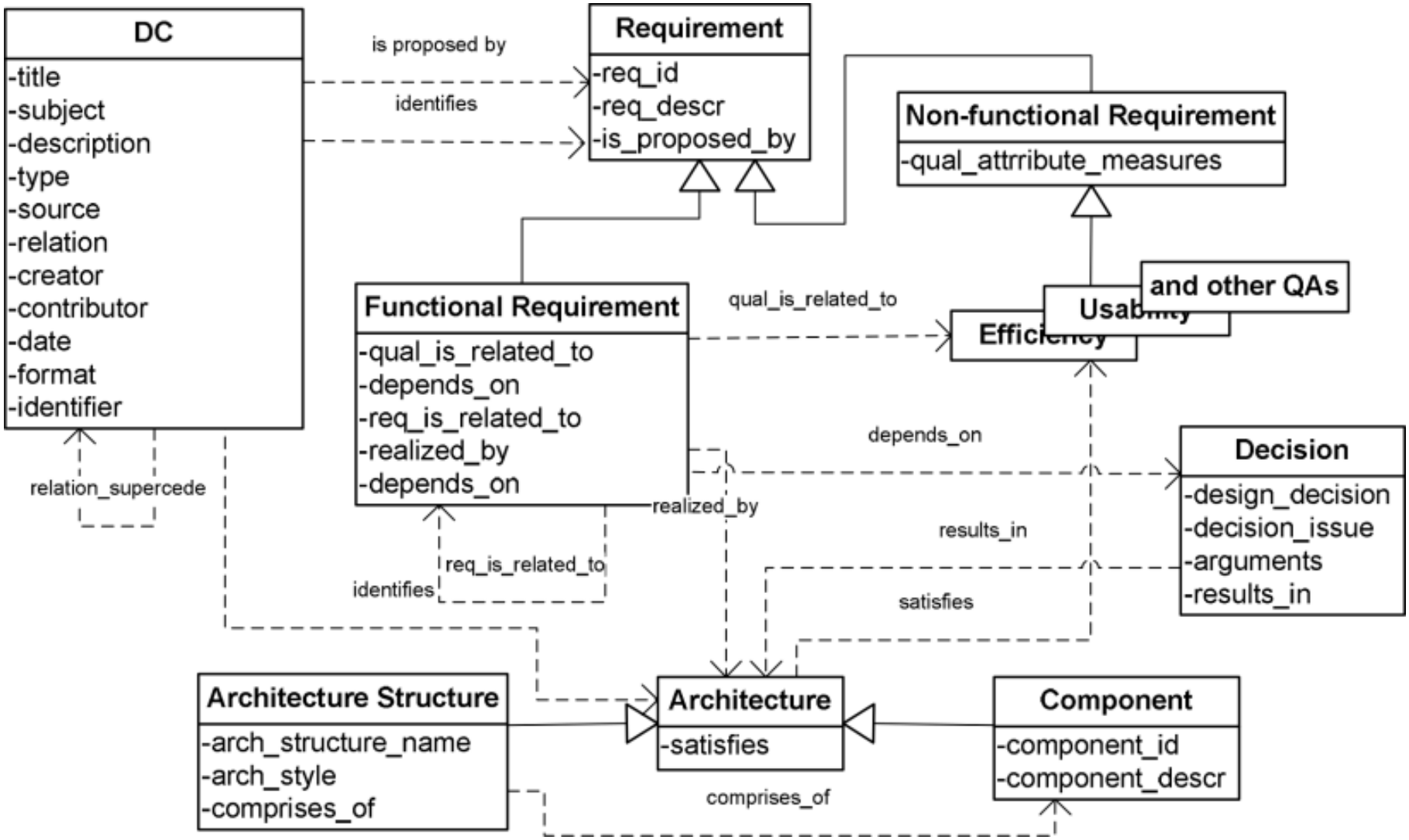


图10. A lightweight ontology for indexing knowledge in requirements and architecture design documents

Antony Tang, **Peng Liang**, Hans van Vliet **Software Architecture Documentation: The Road Ahead.** 2011 Ninth Working IEEE/IFIP Conference on Software Architecture(WICSA 2011), 20-24 June 2011.

案例2：Consistency Checking BTW SA and SAD

- An overview of our plan to realize our idea is depicted in Fig. 11 and can be divided into three major parts:
  - (I) creation of a knowledge base,
  - (II a) pre-processing of the input SAD texts followed by
  - (II b) the main processing to generate a thorough understanding of the text, and
  - (III) checking the consistency between the input documents and the system's architecture in the post-processing step.

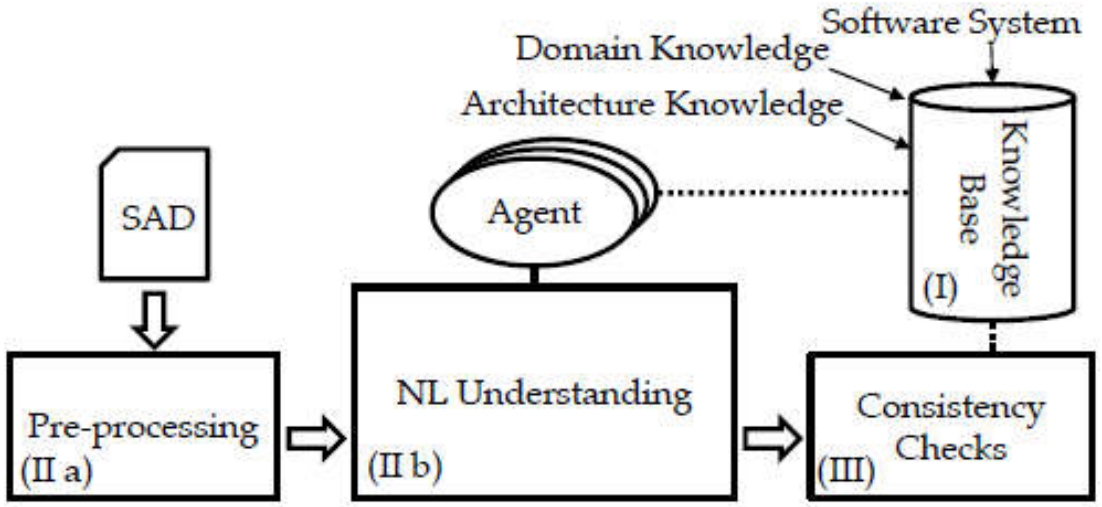


Fig. 11. Overview of planned approach

Jan Keim, Anne Koziolek. Towards Consistency Checking between Software Architecture and Informal Documentation ICSA 2019

案例3 ： A Combined Method for Usage of NLP Libraries Towards Analyzing Software Documents

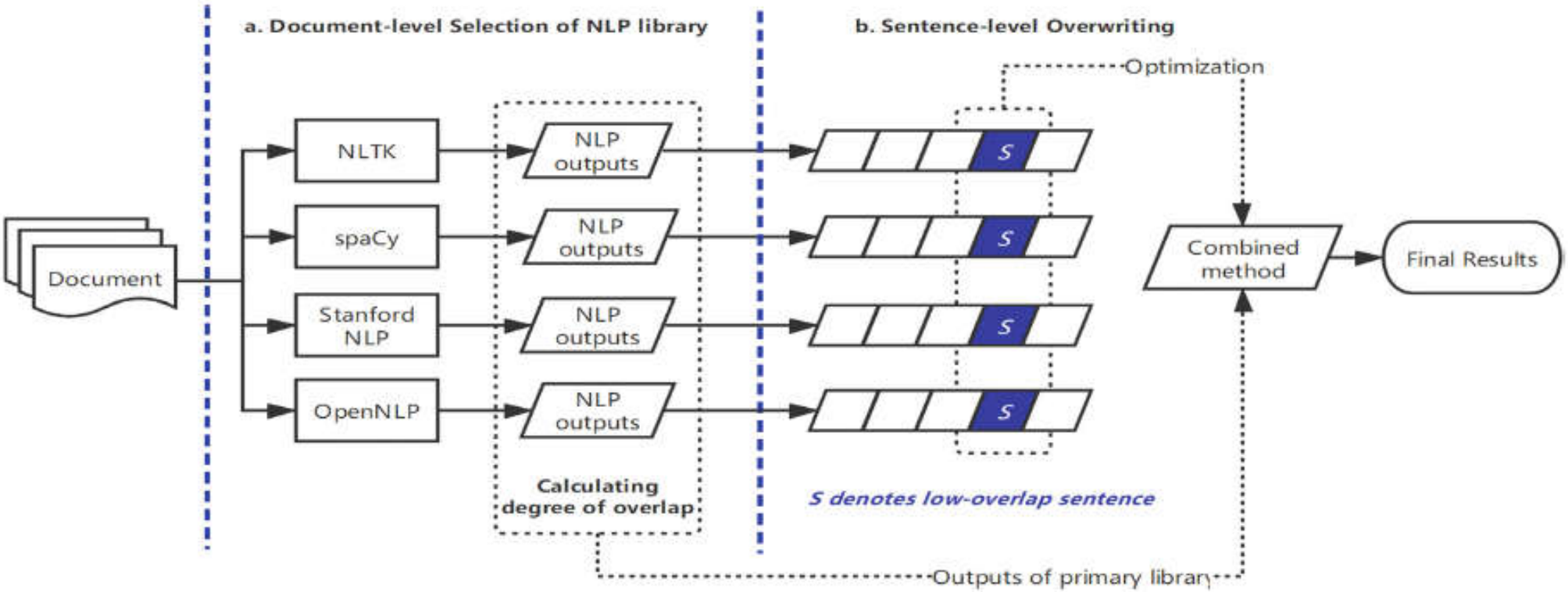


Fig.12. Overall framework of the combined method

Xinyun Cheng, Xianglong Kong, Li Liao, **Bixin Li**. *A Combined Method for Usage of NLP Libraries Towards Analyzing Software Documents,* CAiSE 2020: 515-529.

④ AI+SA评估

- 输入
  - 架构文档
  - 评估要求：质量需求、功能需求、其他需求
  - 评估方法：**ATAM**-Architecture Tradeoff Analysis Method
- 输出
  - 评估结果
- 参与人
  - Stakeholders
  - Outside evaluators

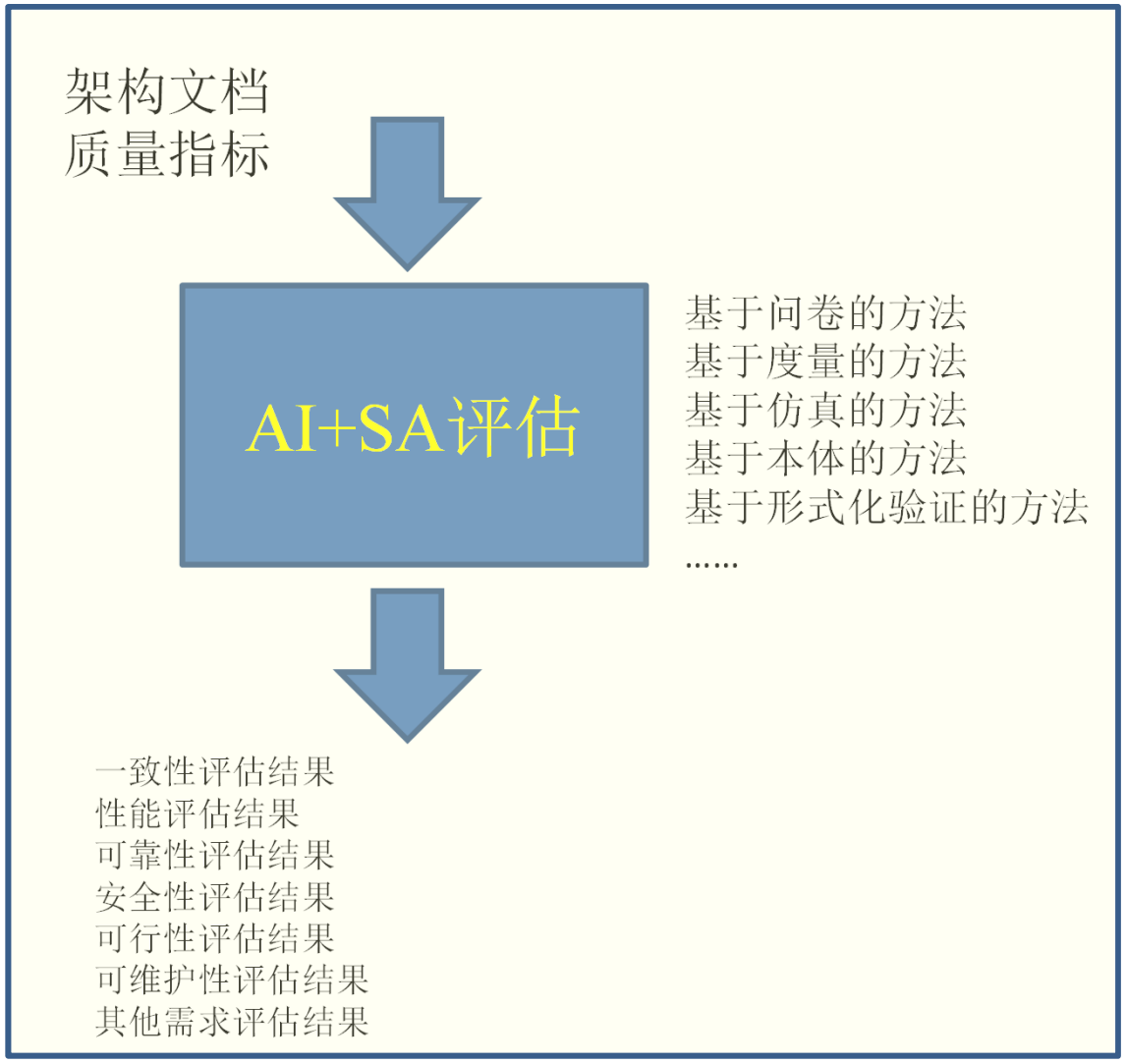


Fig. 13. 架构分析和评估

案例1：Semantic Architecture Workbench

- A proposed Semantic Architecture Workbench (SAW) for automated, ontology-based architectural analysis is depicted in Figure 14. It uses a **triple store** as a database for storing software architecture *axioms and facts* as well as other metadata such as *annotations*. It integrates a set of semantic processing tools to manipulate, infer, and render ontological knowledge. The triple store can be implemented with either conventional relational databases or more modern NoSQL technologies such as graph or columnar databases.

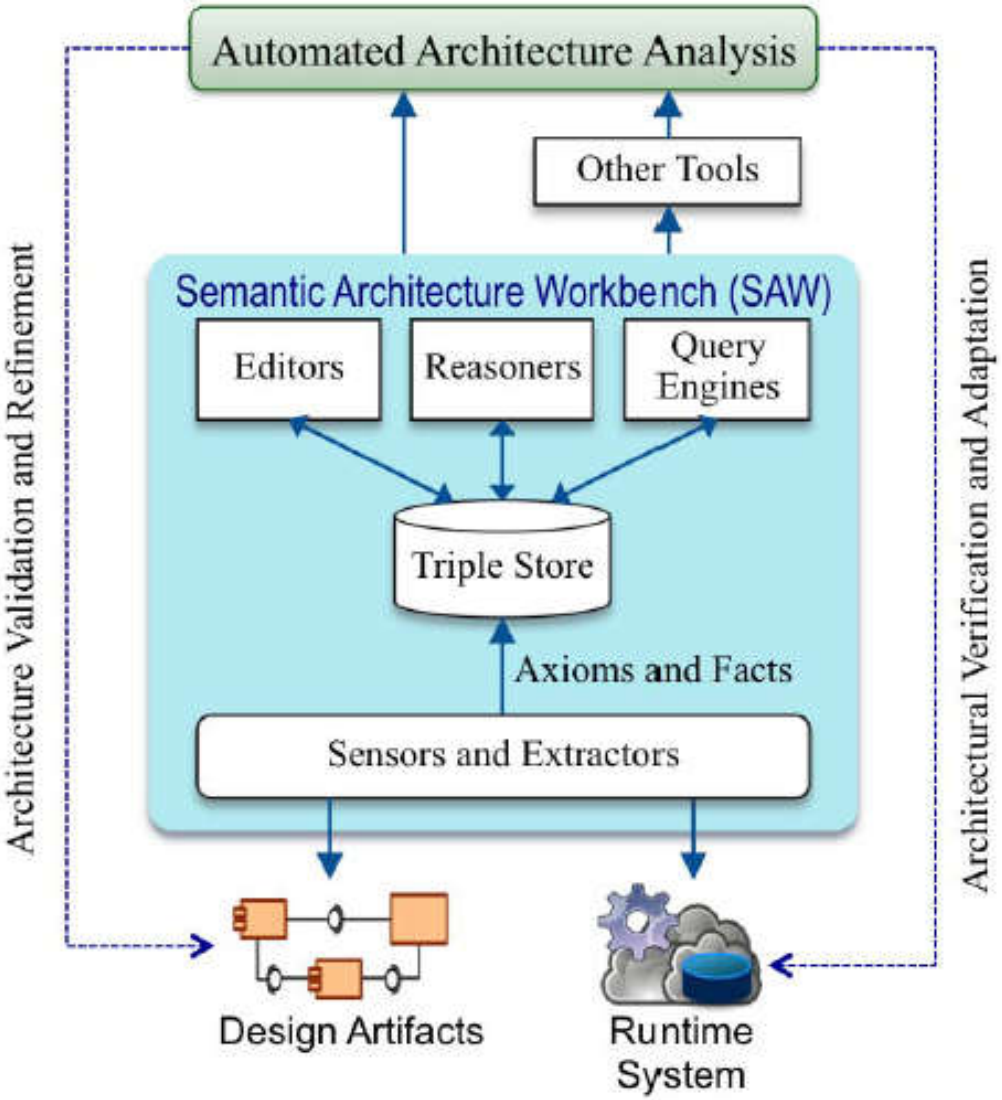


Fig.14. Semantic Architecture Workbench

What is in the Triple Store?

- The SAW also serves as a powerful environment for architectural knowledge reuse and collaboration. As shown in Fig. 15, a project does not need to define its architectural ontologies from scratch, but can import community and domain-specific ontologies already available from elsewhere.
- For example, the project can import **an ontology for the C2 architecture style** and **a domain ontology on TCP/IP networks** before starting to model a distributed system, the same way a Java programmer imports 3rd party packages in their code. Overtime, the triple store is enriched with inferred knowledge from reasoners as well as extracted facts from design artifacts and running software, keeping the ontology repository up to date with the evolving system.

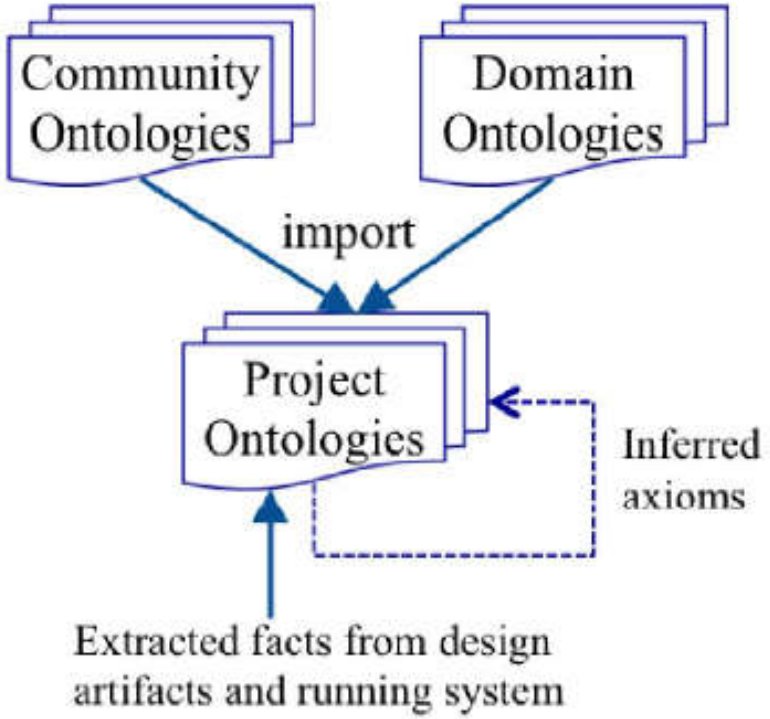


Fig.15. What's in the Triple Store?

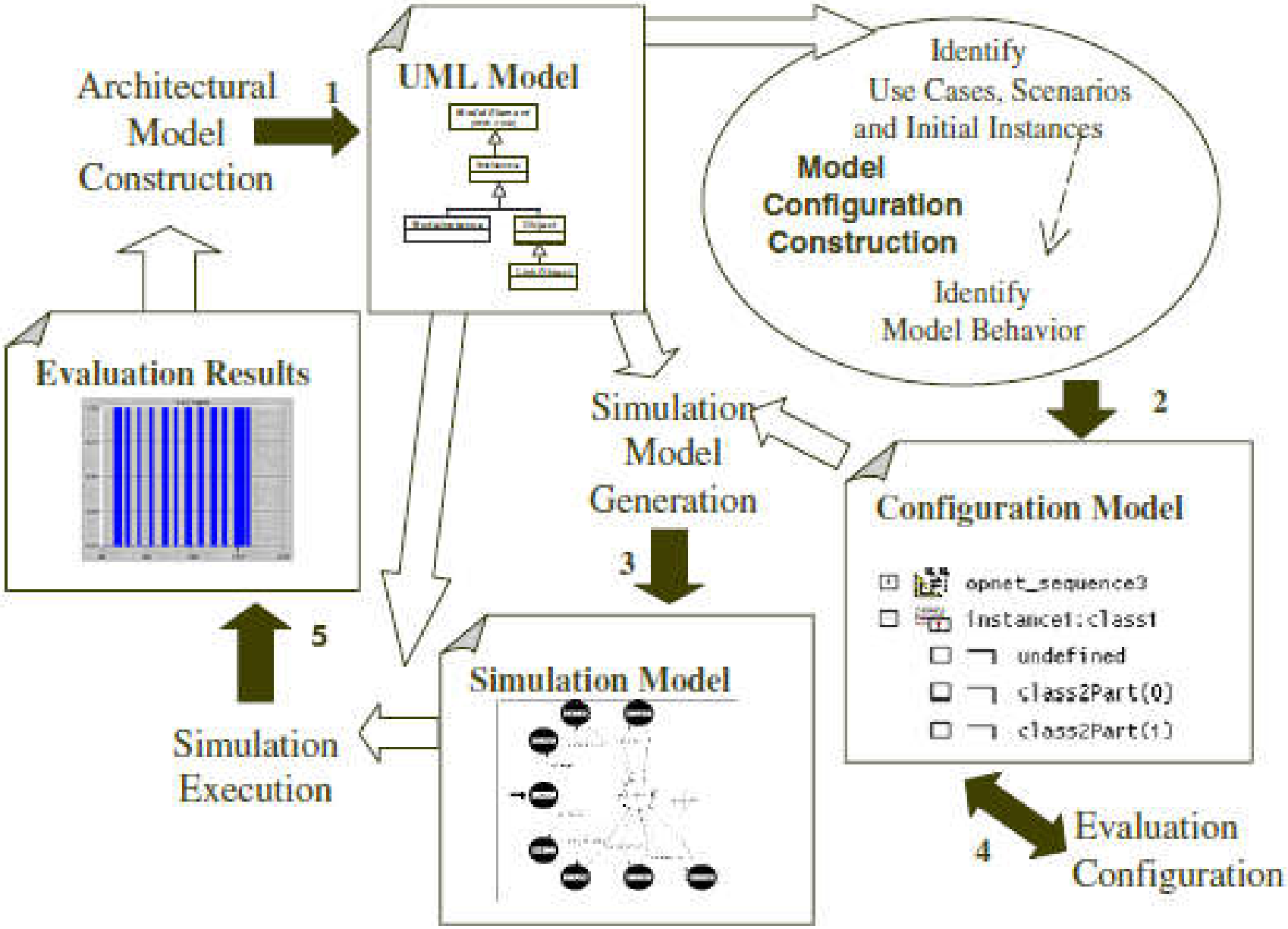


图16.架构仿真的原理和步骤

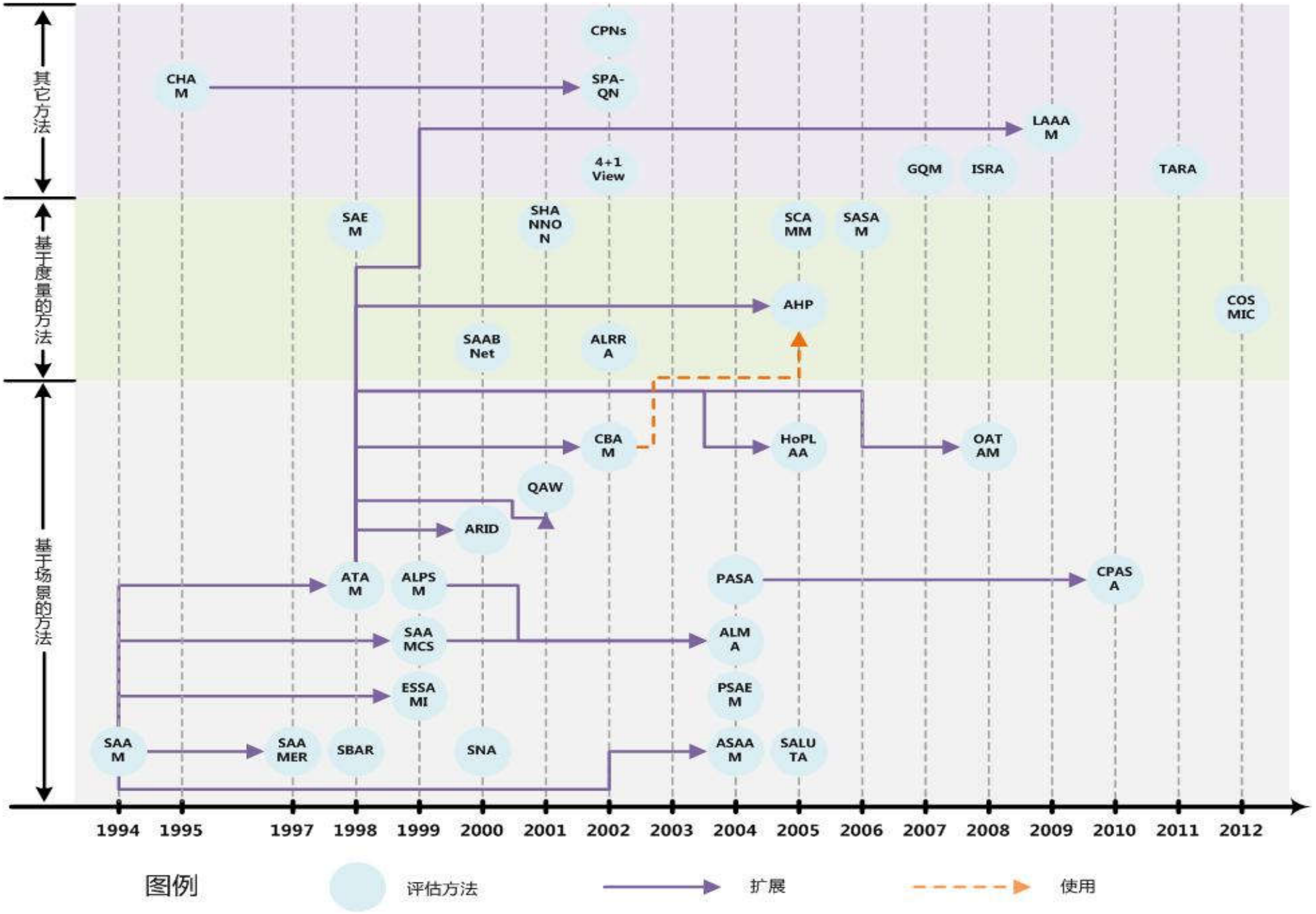


图16. 软件架构评估方法发展路线图

⑤ AI+SA实现

- 输入
  - 设计模型[ADL、UML、其他]
- 输出
  - 架构实现代码
  - 模块实现代码
  - 接口实现代码
- 参与人员
  - 编程人员
  - 测试人员
  - 客户

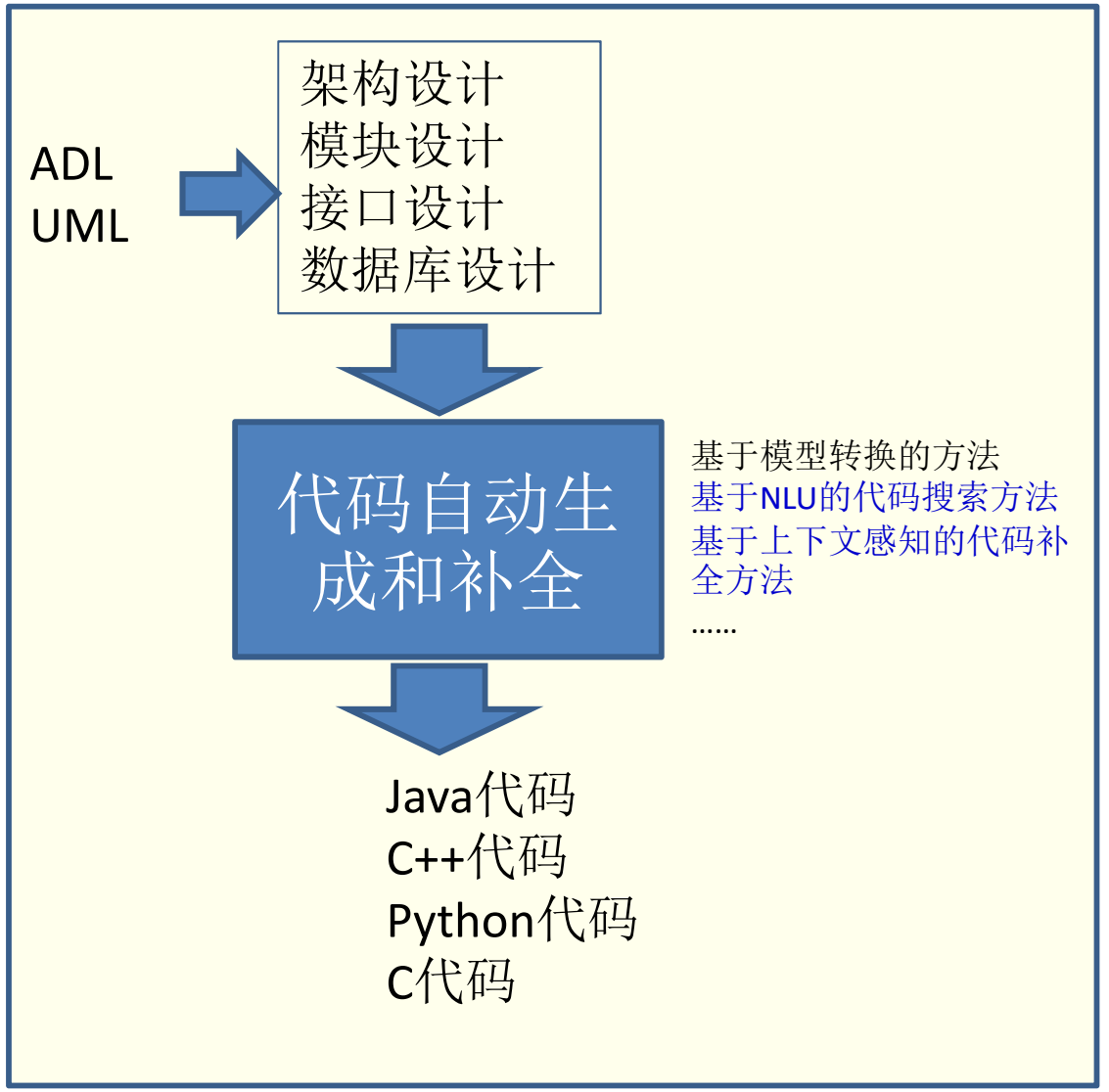


图17.代码自动生成或补全

案例1：HiRec: API Recommendation using Hierarchical Context

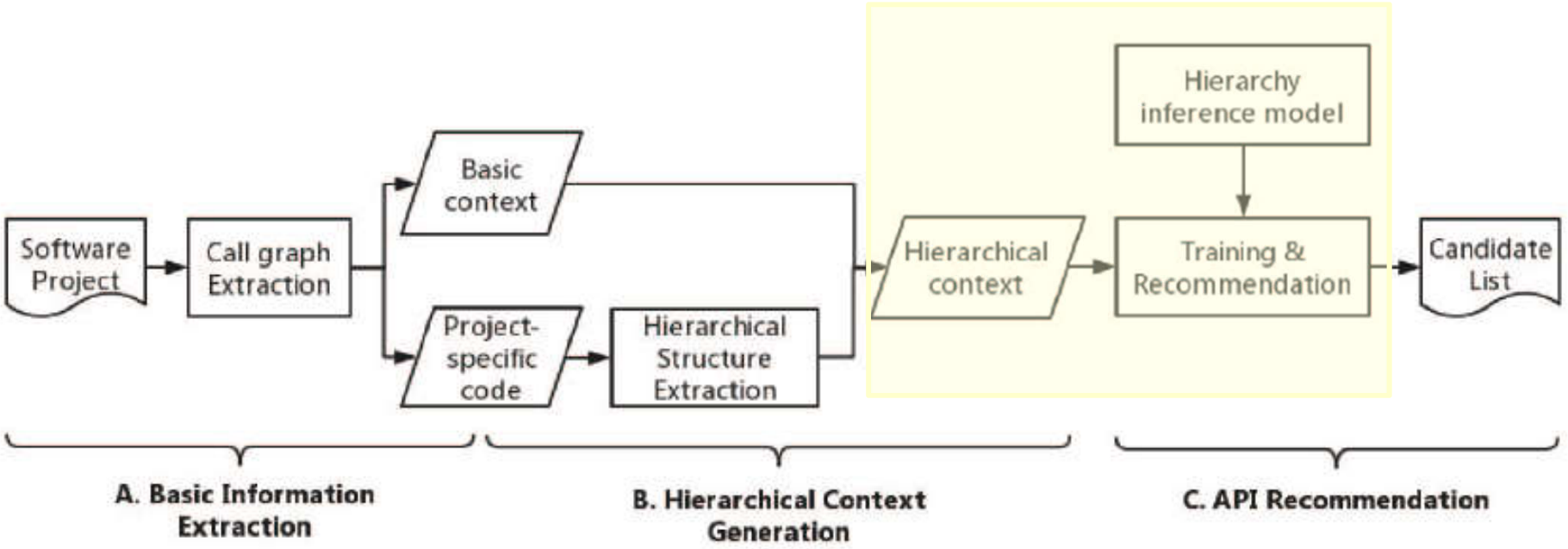


Fig.18: Overall framework of **HiRec**

Rensong Xie, Xianglong Kong, Lulu Wang, Ying Zhou, Bixin Li: HiRec: API Recommendation using Hierarchical Context. ISSRE 2019: 369-379

案例2： Analysis of Correctness for API Recommendation

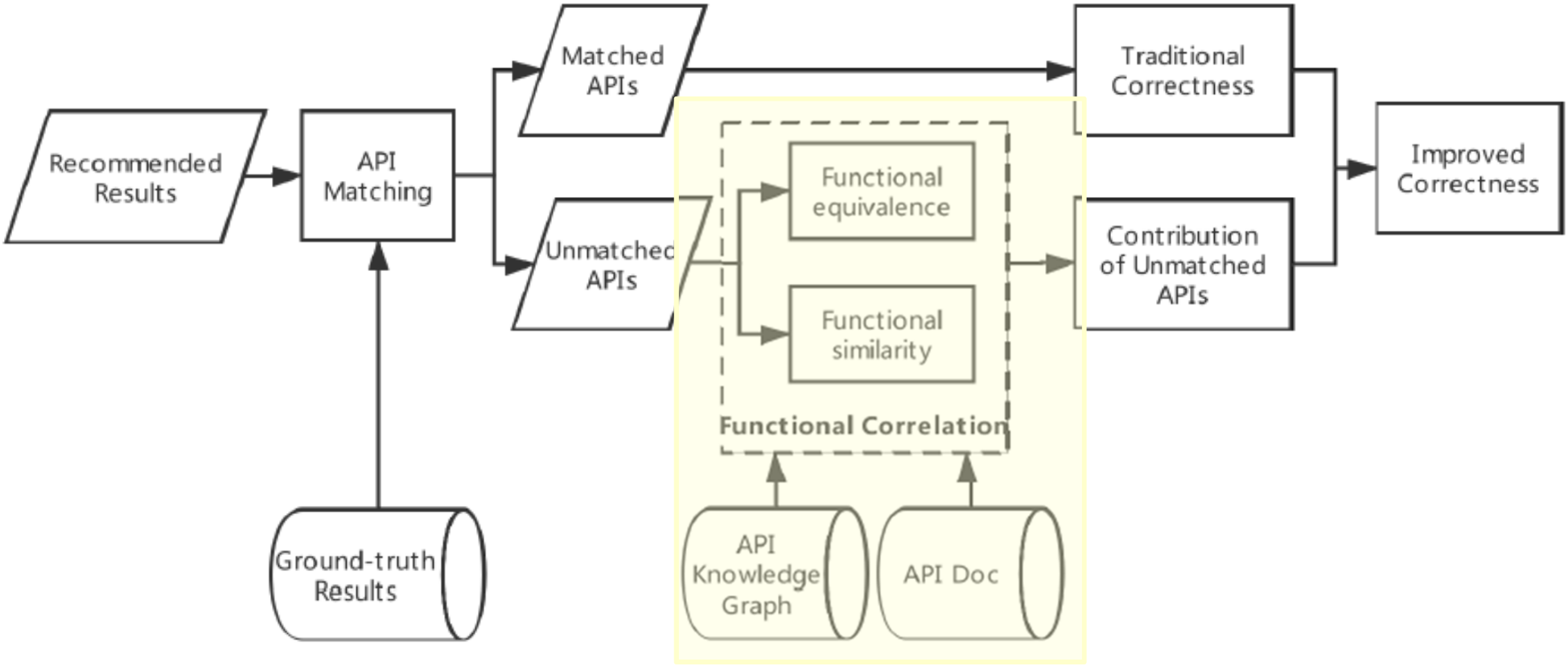


Fig 19. The overall framework of the evaluation of correctness based on API functional correlation

Xianglong Kong, Weina Han, Li Liao & Bixin Li. An Analysis of Correctness for API Recommendation: Are the Unmatched Results Useless? SCIENCE CHINA Information Sciences, 2020.

⑥ AI+SA维护

- 输入
  - 纠错性修改需求：改错
  - 完善性修改需求：增加新功能、提高性能
  - 适应性修改需求：适应新环境
  - 预防性修改需求：提高可扩展性
- 输出
  - 修改后的SA、修改影响分析CIA报告
- 要求
  - 保持一致性
  - 质量保障

方法2：基于AI的软件架构自动生成、评估、演化和优化

- **利用机器学习、知识库、模式库等等进行自动化的软件初始架构生成、评估、演化和优化等。**

- **核心思想和一般过程**
  - ① 基于机器学习的初始软件架构自动生成
  - ② 基于学习和推理的软件架构知识库构建
  - ③ 基于知识库的软件架构评估、决策、演化和优化

① 基于机器学习的软件架构自动生成

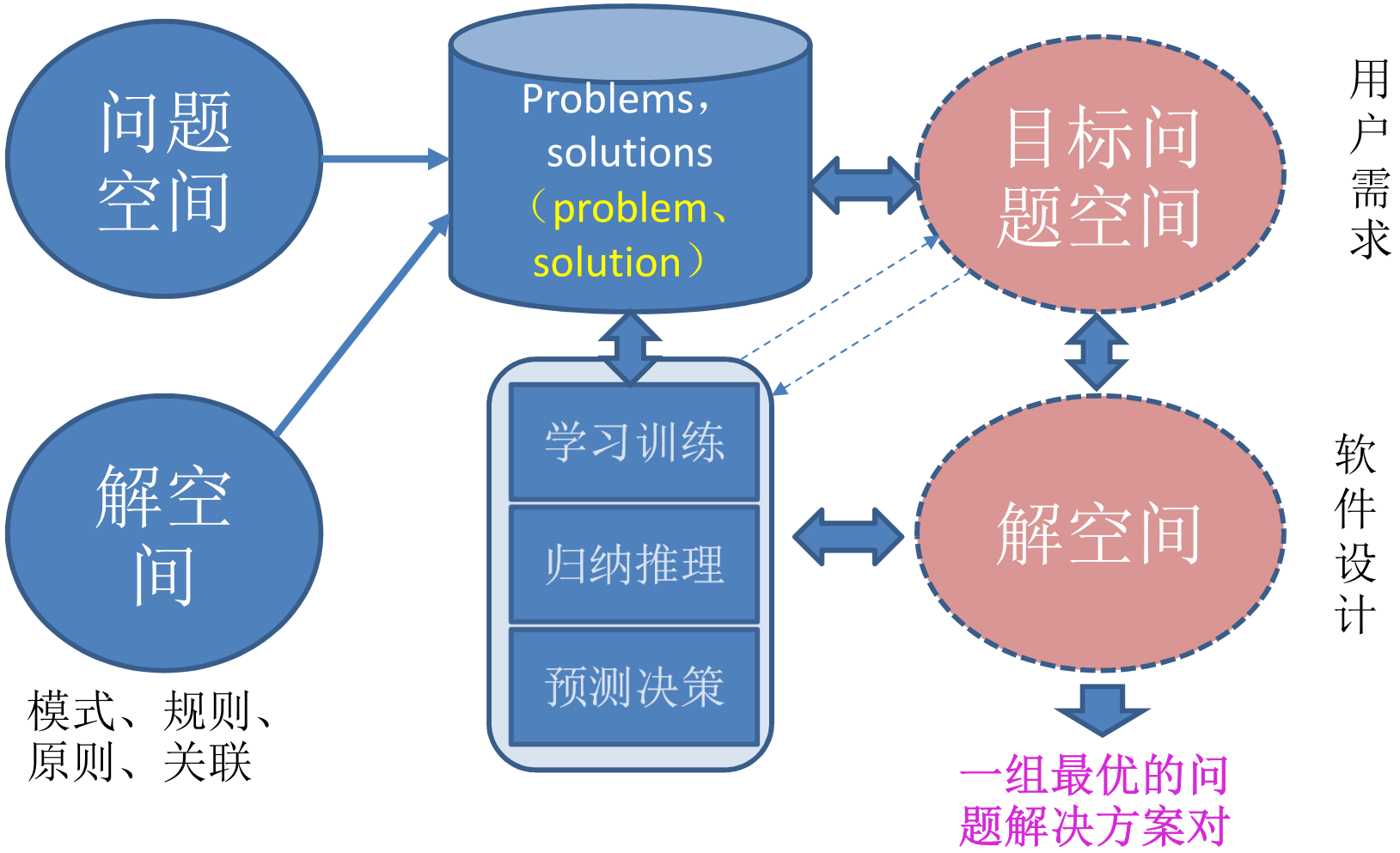


图20. 软件架构自动生成

- 有效处理不确定需求；即时适应运行环境的变化；构建最优(或最合适)软件架构

② 基于学习和推理的软件架构知识库构建

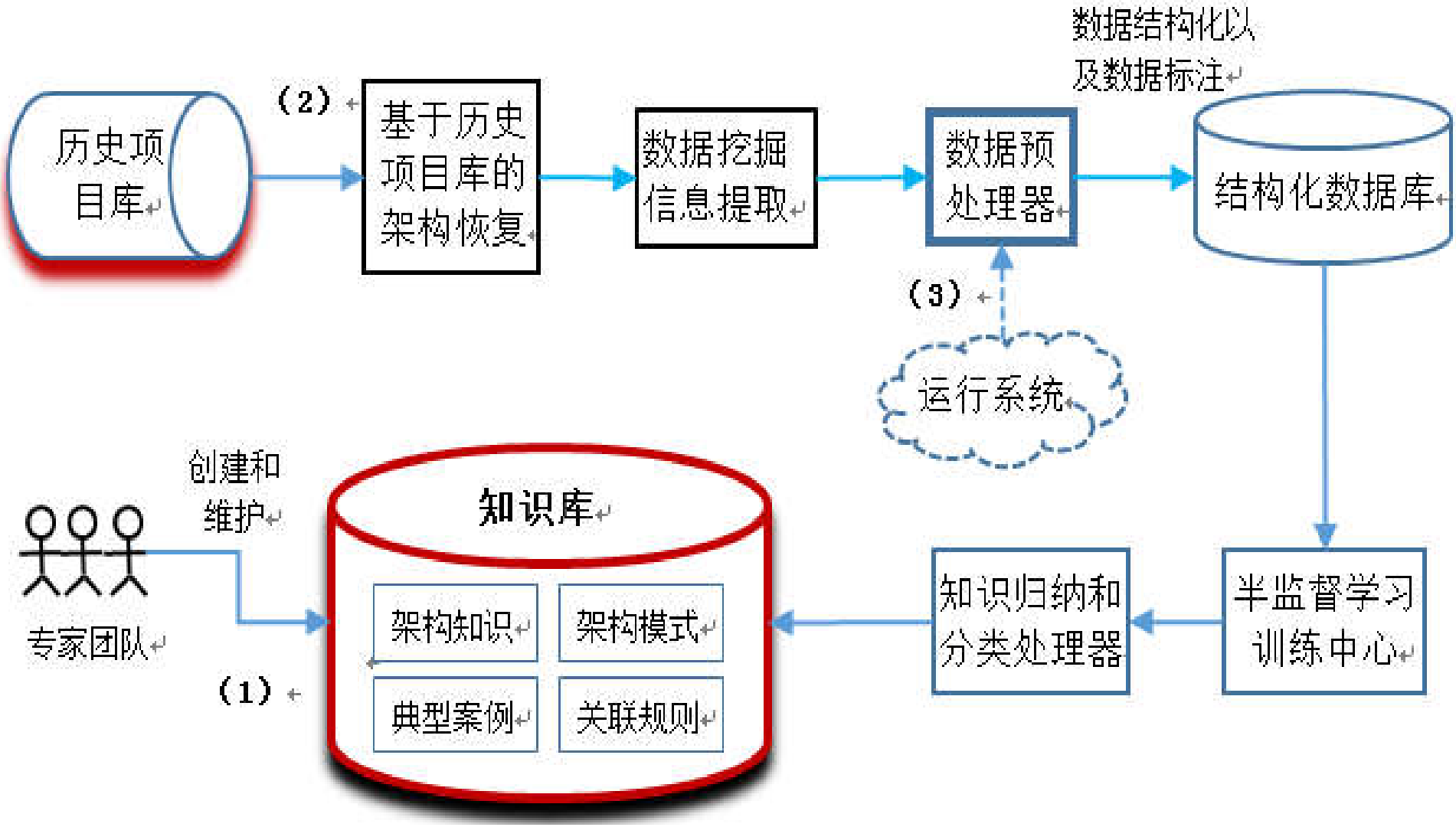


图21. 知识库构建过程

③ 基于知识库的软件架构评估、决策、演化和优化

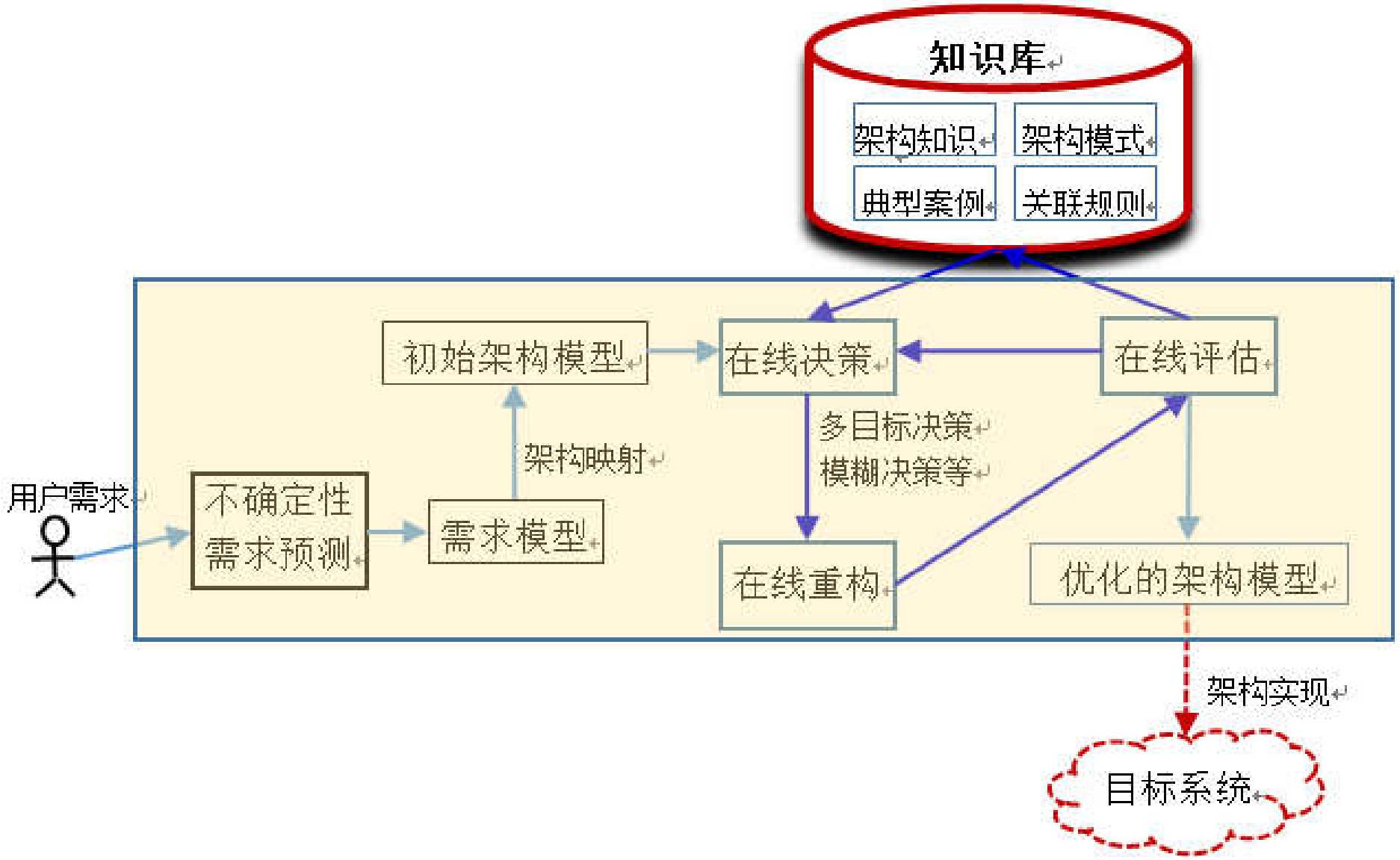
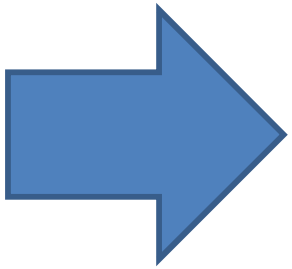


图22. 软件架构在线评估、决策、演化和优化

2.2 SA for AI-based System

- **SA自身具有智能，则SA应该至少具备一定的如下能力**：
  - 感知能力
  - 学习能力
  - 推理能力
  - 预测能力
  - 决策能力



情况一：SA是由一些智能体（例如Agent）组成的，也就是说组件（component）、连接件（connector）都是智能体，具有感知、学习、推理、预测和决策能力。

情况二：SA的组成元素本身不是智能体，但SA自带智能助手（SAIasistant）， SAIasistant相当于架构师，具有感知、学习、推理、预测和决策的能力，但比架构师似乎更强大。

情况三：上述情况的混合

最终的**ISA**具有：（1）环境感知、自演化、自修复、自优化等自适应能力！(self-adaptive software architecture)；（2）Optimal SA

案例1：无人驾驶

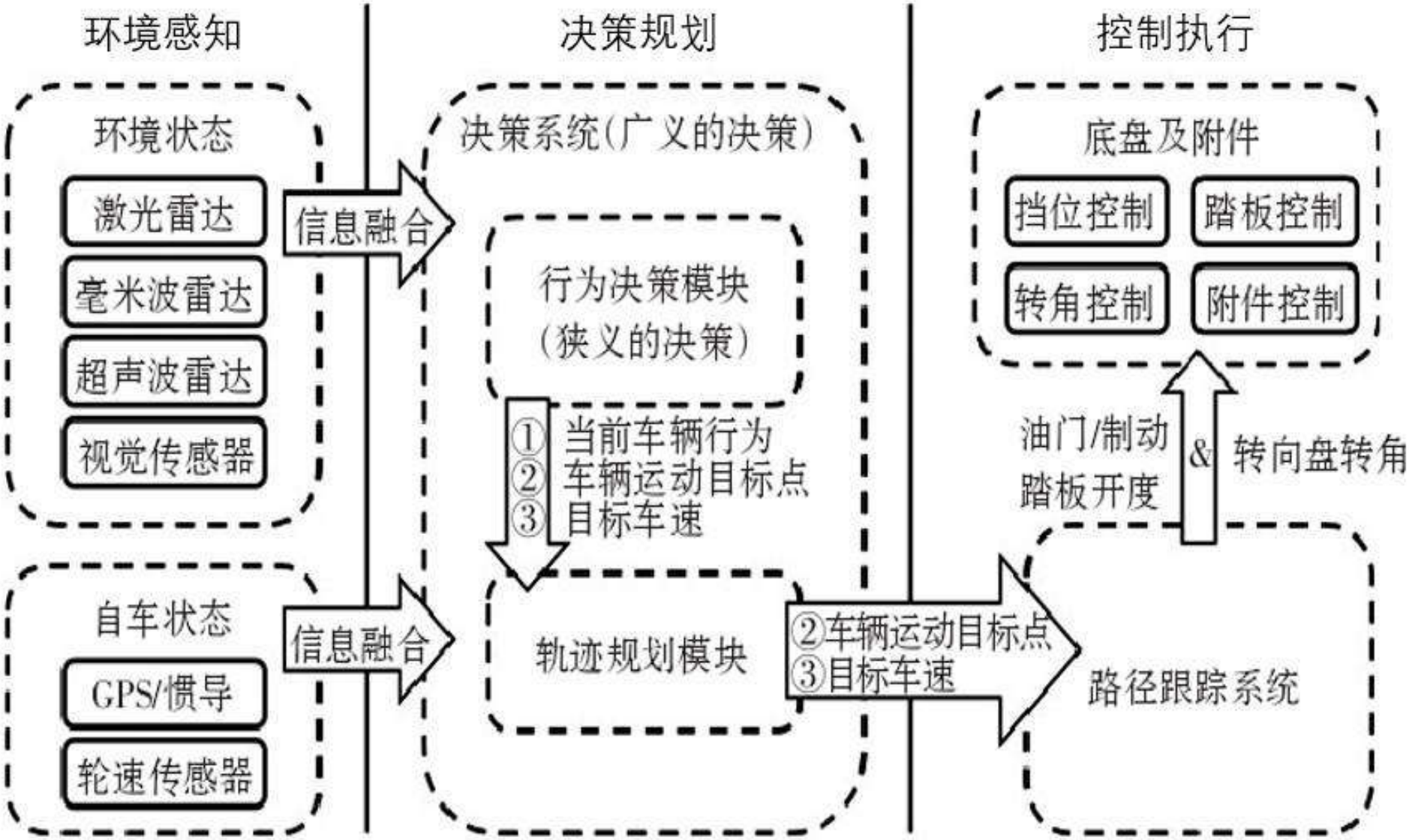


图**23**. 无人驾驶系统架构

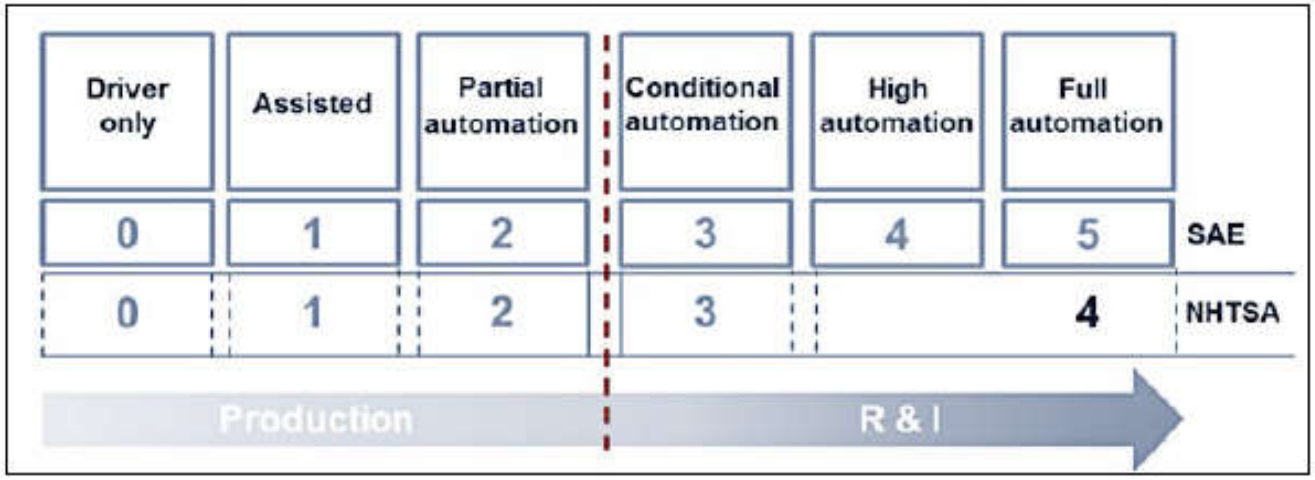


Fig.24. Level 0-5 from no to fully autonomous car with no driver in loop

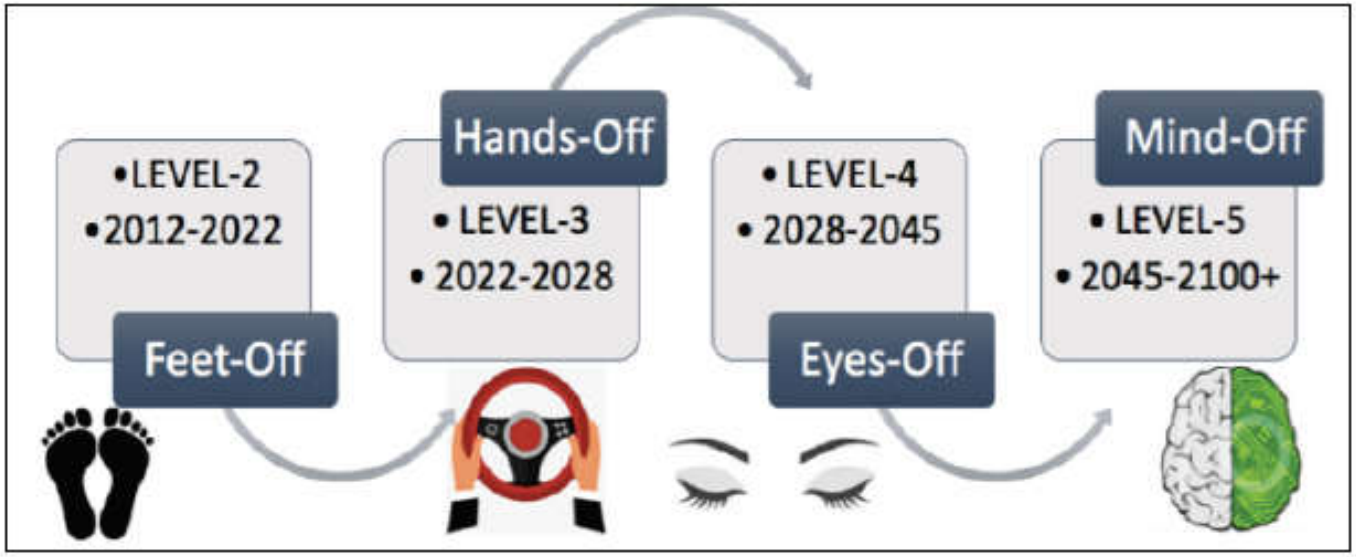


Fig.25. Vehicular intelligence pathway will lead to less human intervention in coming years.

案例2：智能交通远景

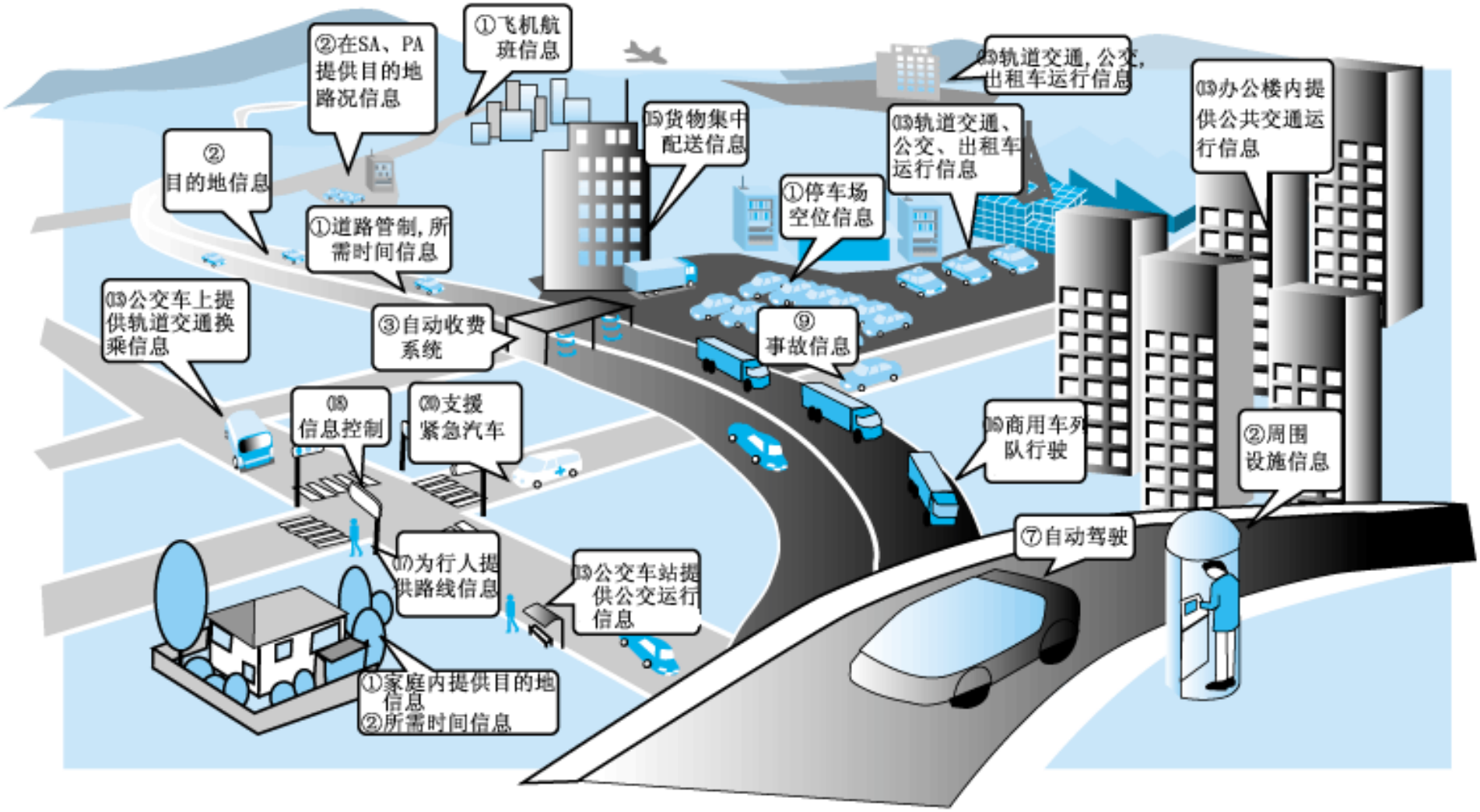


图26. 智能交通远景

智能交通系统架构：（来自公安部交通管理科学研究所研究员 邱红桐的报告）

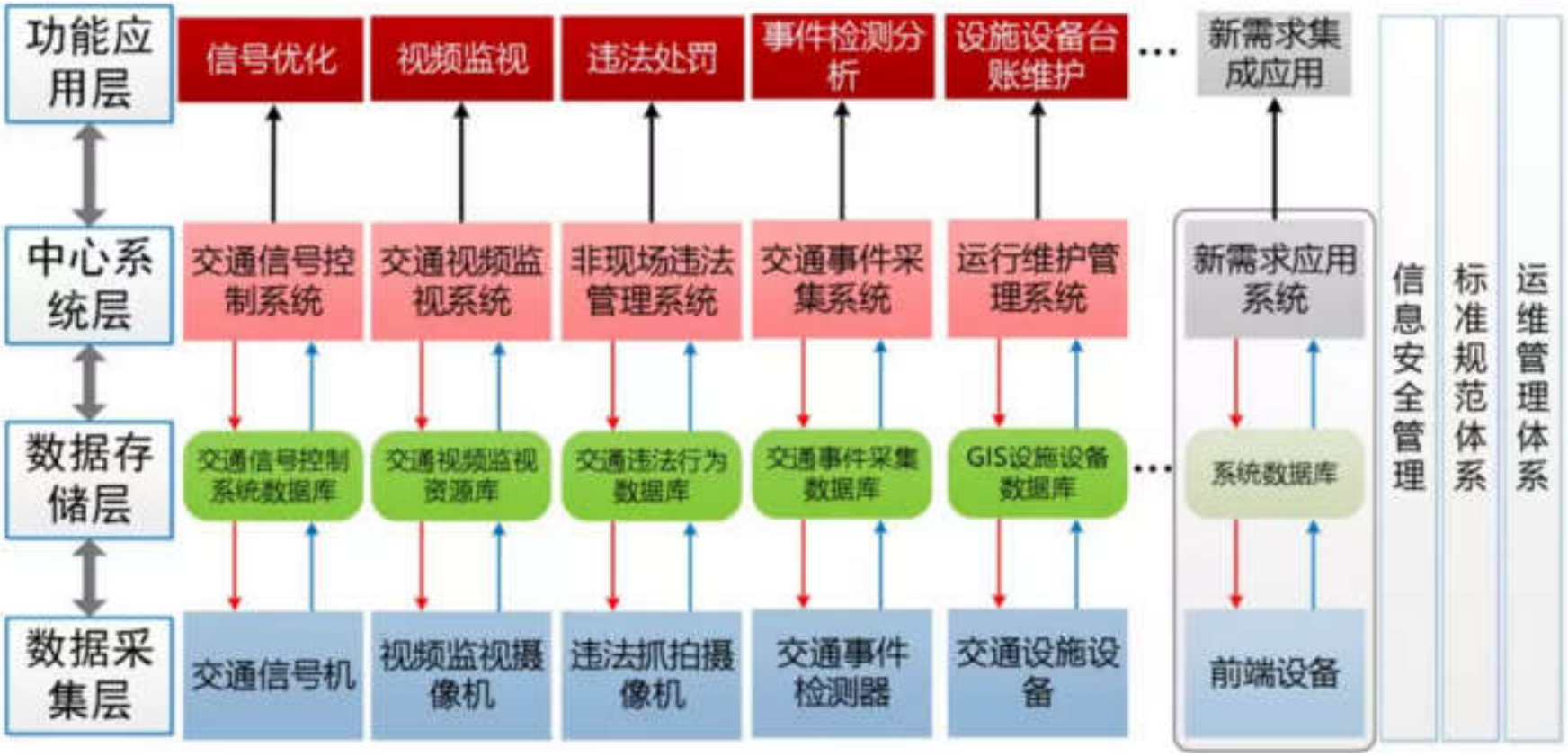


图27. 发展初期的“烟囱式”架构

初期为“烟囱式”架构系统林立，设备间没有联动控制，系统内部没有数据交互。近期为“集成式”架构统一系统数据接入，集成各子系统应用。

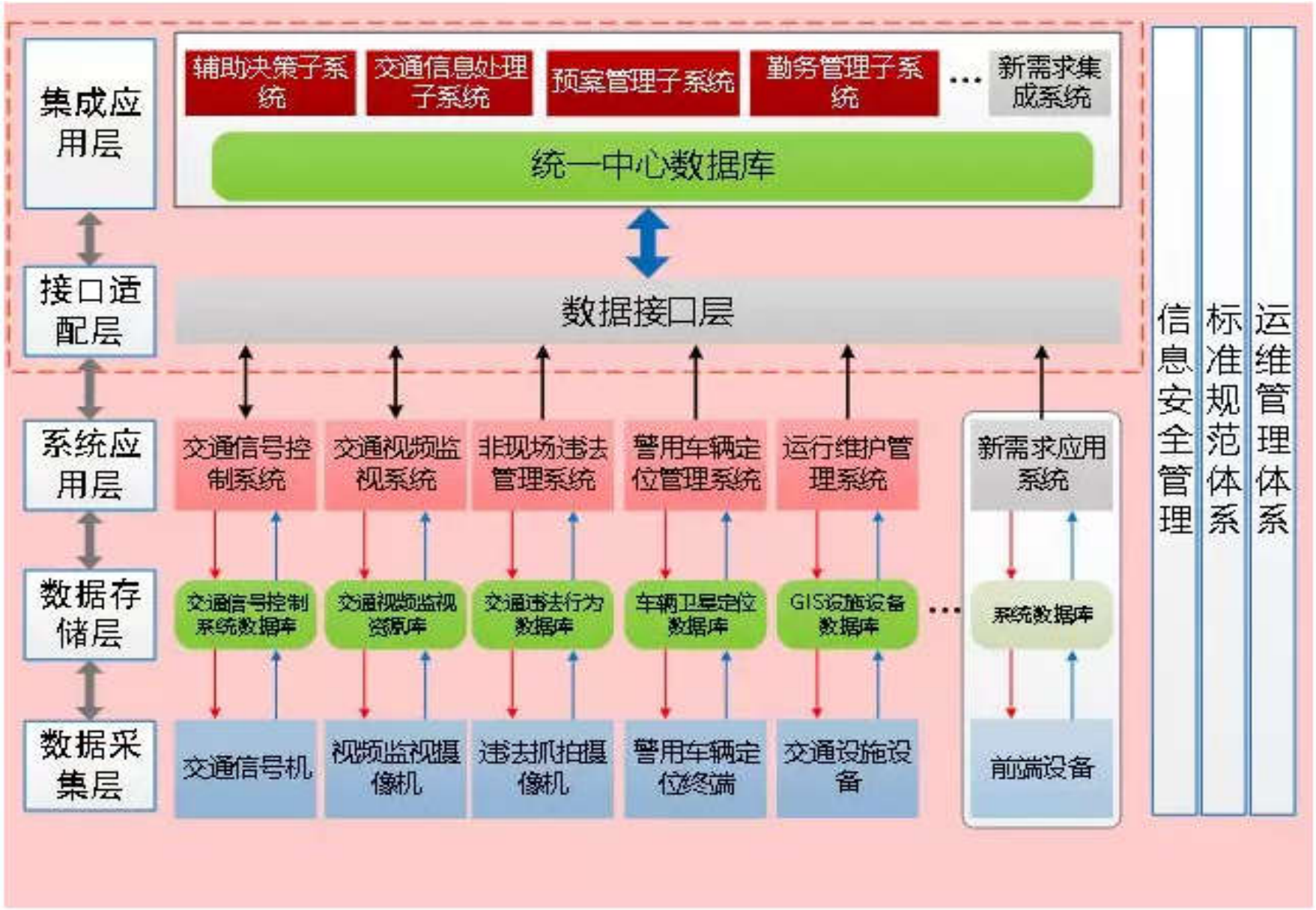


图28. 发展到近期为“集成式”架构

•第一个是业务应用层，也就是我们说的用户交互层，这是系统面向用户最直接的层面，当前和今后一段时期呢，我们整个系统对路面管控的业务重点是“情、指、勤、督”四位一体。不管是设计阶段还是开发运营阶段，在整个系统的生命周期过程中，用户的应用需求会不断的变化和提高，只有变才能有生命力。我们希望这块能向轻量化和定制化的方向转变。

•第二个是应用支撑层，这个就是对上一个应用层技术支撑，这层建立主要是为了实现业务应用层轻量化，定制化提供支撑。

•第三个是数据共享层，也是我们的数据资源层，这一块应该说是行业讨论最热烈的，阿里也好，华为也好，很大程度上都是讲这一块功能。这里包括三个方面，一个是按照统一的数据采集接口，实现内外部数据种类丰富多样的统一，包括清洗校正这些，最主要的是数据，包括信号控制数据，视频数据等等，把这些数据统一接入进来。

•第四个是中心平台层，构建应用软件开发所需中间件的统一管理环境，向应用软件提供中间件服务，促使业务应用系统软件的开发、部署及发布变得快捷、高效，软件环境资源可实现集约化应用。

•第五个是基础设施层，将各类异构、迅捷、强大的计算资源、存储资源、网络资源资源池化，实现资源的统一集中管理，充分发挥云平台虚拟化计算、按需使用、动态扩展的特性。

•第六个是感知、认知层，感知技术将向“N合一”多功能复合型、集约型方向发展，实现前端赋能+后台智能识别，从前端感知提升到前端认知。

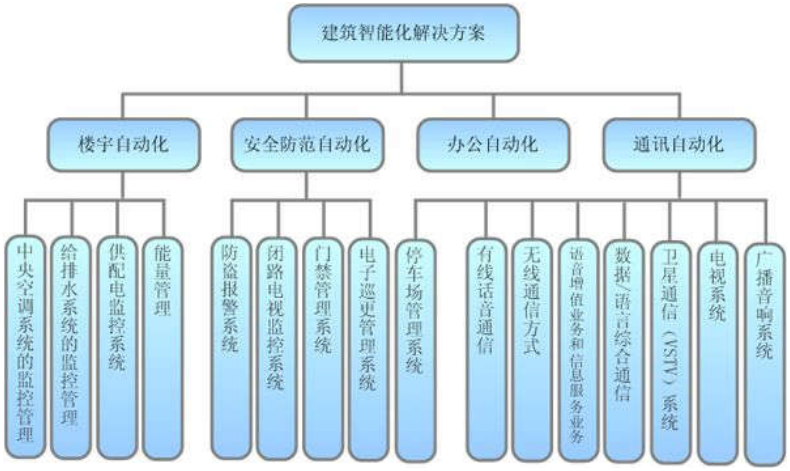


图29. 智慧城市和智能建筑

三、智能化软件架构面临挑战

- **数据集的完整性和正确性：** 正负样本空间的局限性，训练学习对象的具有多样性吗？
- **知识库的有用性：** 机器学习算法的学习能力怎么样？学习到的知识的有用吗？
- **学习模型的有效性：** CNN、朴素贝叶斯、集成学习？
- **预测模型的有效性：** 预测能力、预测效果怎么样？
- **决策模型的有效性：** 各种决策模型的决策效果如何？模糊决策模型，多目标决策模型，随机决策模型，决策树.
- **支持软件架构构建过程各种工具的自动化、智能化程度如何？**

四、智能化软件架构存在机会

•4.1 在保障数据集的完整性、正确性、时效性、可信性方面

–历史数据[类似历史项目]

- 文档、模型、数据、代码、算法
- 依赖、关联、日志、关系
- 规则、规律、模式、原则、知识

–专家数据[架构师/领域专家/人工智能专家]

–开发数据[开发过程数据]

–运维数据[运行过程数据]

•4.2 在保障知识的有用性、可信性方面

–知识类型多样性

- 架构知识：领域架构知识图谱
- 架构模式/设计模式：问题解决方案对
- 典型案例：该领域典型成功案例（正样本）、典型失败案例（负样本）
- （架构）设计原则：职责单一
- 关联规则：制品关联
- 知识网络、知识本体、知识图谱

–知识获取途径

- 机器学习：强化学习/深度学习
- 知识挖掘：各种挖掘方法
- 知识推理：

•4.3 在保障预测模型的有效性方面

–预测能力

–预测效果

–常见预测方法的使用

- 定性预测方法
- 实践序列分析
- 因果关系预测
- 组合预测
- 其他预测

•4.4 在保障决策模型的有效性方面

–决策能力

–决策效果

–常见决策方法的使用

- 随机决策
- 模糊决策
- 多目标决策
- 组合决策
- 其他决策

•4.5 支持软件架构构建过程各种工具的自动化智能化程度

–架构需求获取工具

–架构设计工具

–架构文档化工具

–架构度量评估工具

–架构测试工具

–架构实现工具

–架构恢复工具

–架构优化工具

–……

感谢您的关注！
感谢邀请！