Performance-Oriented Software Architecture Engineering: an Experience Report

Chung-Horng Lung, Anant Jalnapurkar, Asham El-Rayess
SEAL - Software Engineering Analysis Lab
Nortel Networks

Software Architecture Analysis

- **Established in 1995**
- **Adopted SAAM (Software Architecture Analysis Method) developed at SEI**
  - **Scenario-based analysis**
  - **Non-functional quality attributes like maintainability, scalability, and etc.**

Why Architecture Analysis?

- Increasing complexity of software systems
- The need to analyze and design systems at higher levels of abstraction
- The demand to reduce maintenance costs for evolution
- Often conducted in an ad-hoc manner

Motivation to Integrate SA and SPE

The main reasons are:

- **Performance issue keep recurring for real-time applications.**
- **Need to demonstrate how to improve quality attributes, especially performance in a systematic approach.**
- **Software architecture and software performance are tightly coupled.**

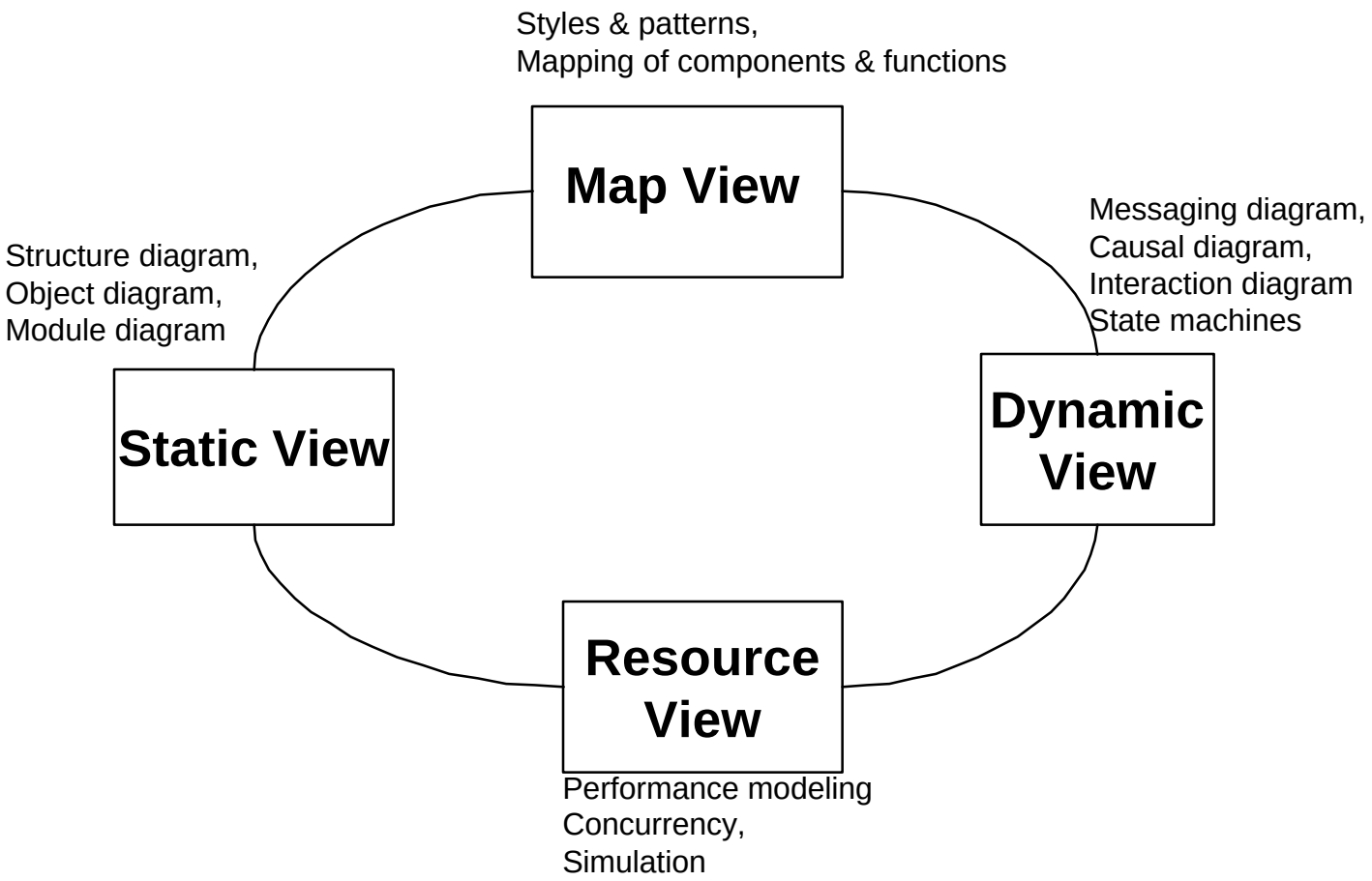
Performance-Oriented Software Architecture Engineering

Performance-Oriented SW Architecture Eng

Critical elements of POSAE:

- **SAAM**
- **Stakeholders and their values**
- **Architectural views**
- **Software partitioning and clustering**
- **Software performance engineering**
  - **Automatic generation of performance models**
- **Software architecture trade-off analysis**

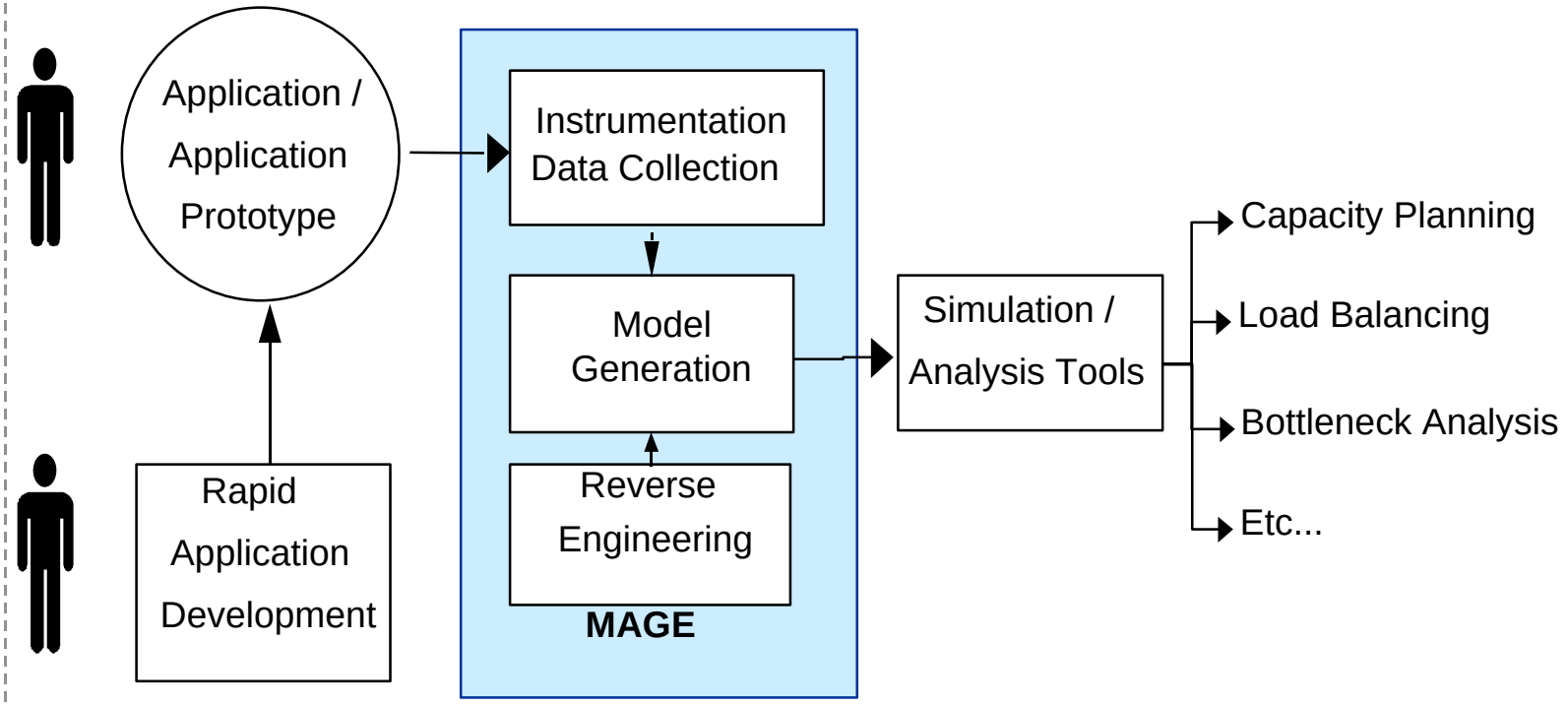
Software Architecture Views



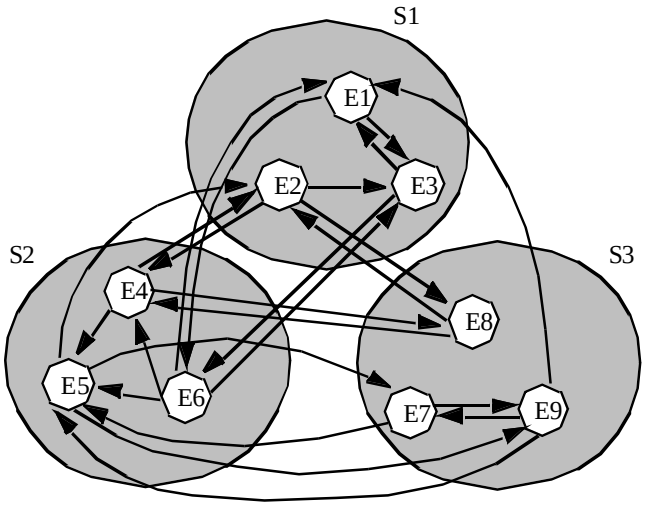
POSAE Process - Iterative & Incremental

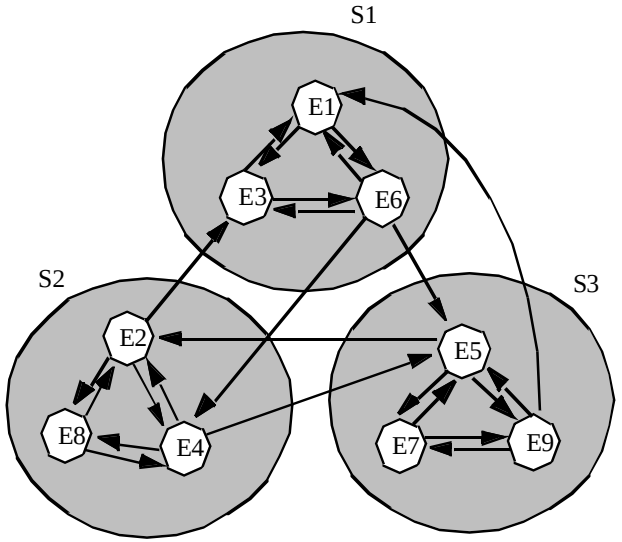
- **Develop or capture a software architecture** (static view)
- **Identify scenarios, particularly real-time (RT) scenarios** (scenario development)
- **Identify execution paths for RT scenarios** (dynamic view)
- **Apply performance modeling, analysis, and measurements** (resource view)
- **Perform architecture analysis based on performance modeling results** (map view, dynamic view, and resource view)
- **Conduct trade-off analysis** (scenario & 4 views)
- **Build a prototype, based on the analysis, to improve performance or other qualities**

Performance Modeling & MAGE



Software Partitioning: an Example





Lessons Learned

- **End-to-end analysis provides valuable insights.**
  - **Messaging system, run-time environment, application framework, and the high-level services and applications.**
- **Software architecture is a critical asset & important to SPE.**
  - **Need an engineering approach.**
- **Analysis of the interactions of scenarios, not just individual scenarios, are necessary.**
  - **Example, scenarios query processing, update processing, and OS scheduling interact.**
- **Automation of performance model generation and analysis is needed.**
- **Prototyping is useful to show values & alternatives.**
- **Domain knowledge plays a critical role.**

Conclusion

Presented a POSAE approach

Some benefits and achievements:

- **Capture software architecture**
- **Identify use case scenarios**
- **Improve performance.**
  - **Examples: 25% for one project & 500% for another one**
- **Perform modeling & analysis at the early stage.**
- **Better document the system.**
- **Support product evolution.**

Ongoing Works & Challenges

- **Tool supports**
  - **Reverse engineering tools, especially for OO software**
  - **Reliable performance measurements**
  - **Performance modeling and analysis to support integration of SPE and software development**
- **Design patterns and performance**
  - **characterization of design patterns**
- **Development of best practices and design guidelines**