1. Software Performance Engineering

Connie U. Smith, Ph.D.
PO Box 2640
Santa Fe, New Mexico 87504-2640
(505) 988-3811
http://www.perfeng.com/

Overview

1. Overview of SPE
2. Distributed System Models
3. Model Interoperability
4. SPE Research and Education

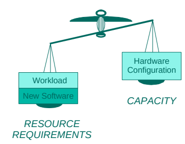
 2

1. Overview of SPE

- Software Performance Engineering (SPE) Introduction
- SPE Process
- Data Required
- Models
- Case Study

 3

Performance Balance



- Quantitative Assessment
- Begins early, frequency matches system criticality
- Often find architecture & design alternatives with lower resource requirements
- Select cost-effective performance solutions early

 4

SPE Focus

- Performance: a measure of how well a software system or component meets its requirements for timeliness
  - responsiveness
  - scalability
- Software-oriented approach
  - focuses on architecture, design and implementation choices

 5

Goal

- Early assessment of software decisions to determine their impact on quality attributes such as
  - performance
  - reliability
  - reusability
  - maintainability/modifiability

- Early assessment of the architecture is important because
  - architecture has the most significant influence on quality attributes
  - architectural decisions are the most difficult to change

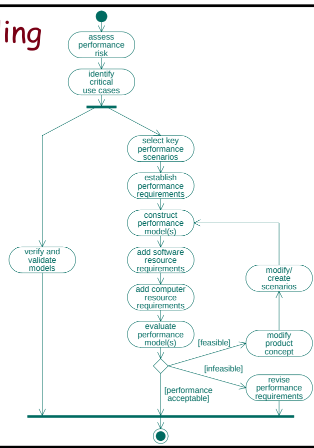
 6

Part 2: Review of SPE Model-based Approach

SPE Process Includes

- General Principles for Performance-oriented Design, Design Patterns and Antipatterns
- Data Gathering Strategy
- Performance Walkthroughs
- Model Construction and Evaluation Strategy
- Techniques to Compensate for Uncertainties
- Data Presentation & Tracking
- Model Verification & Validation

SPE Modeling Steps



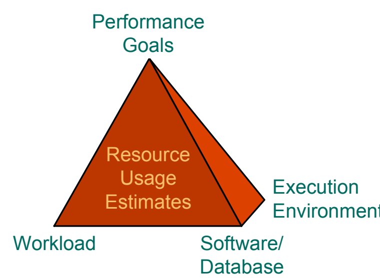
SPE Modeling Steps

1. Assess performance risk
2. Identify critical use cases
3. Select key performance scenarios
4. Establish performance requirements
5. Construct performance models
6. Determine software resource requirements
7. Add computer resource requirements
8. Evaluate the models
9. Verify and validate the models

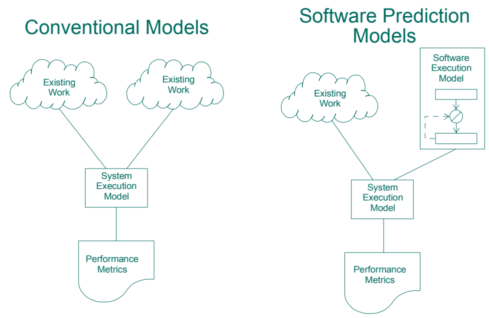
Performance Scenarios

- Workload identification and characterization
- Benchmark scenarios
- Precise performance requirements
  - Response time
  - Throughput
  - Resource budgets

Gather Data



Performance Models



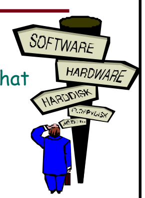
 13

SPE Model Requirements

- Low overhead
  - use the simplest possible model that identifies problems
- Accommodate:
  - incomplete definitions
  - imprecise performance specifications
  - changes and evolution
- Goals:
  - initially distinguish between "good" and "bad"
  - later, increase precision of predictions
  - provide decision support

 14

SPE Modeling Strategies



- *Simple-Model Strategy*
  —start with the simplest possible model that identifies problems with the system architecture, design or implementation plans.
- *Best- and Worst-Case Strategy*
  —use best- and worst-case estimates of resource requirements to establish bounds on expected performance and manage uncertainty in estimates
- *Adapt-to-Precision Strategy*
  —match the details represented in the models to your knowledge of the software processing details

 15

Cost / Benefit Analysis

- Schedule impact
- Modification Effort
- Side Effects
- Risk

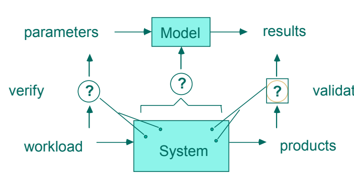
↔ Predicted Improvement

If alternatives are unacceptable, revise the performance requirements

Best alternative may not have the best performance

 16

Model V & V



- Verify the model data, validate the results
- Begin early in the life cycle and continue throughout life cycle
- Check for model omissions!

 17

SPE Model-Based Approach

- Reduces the risk of performance failures
- Provides:
  - Refinement and clarification of performance requirements
  - Predictions of performance
  - Estimates of sensitivity to precision of resource usage estimates and workload intensity
  - Quantitative impact of software alternatives
  - Scalability of the architecture and design
  - Identification of critical parts of the system
  - Identification of assumptions that may change results
  - Assistance for budgeting resource demands
  - Assistance for designing performance tests

 18

Part 2: Data Required

Workload Data

- Pareto principle ('80-20 rule')
  - More than 80% of the software requests will be for less than 20% of the functions of the system
- First: scenarios of *typical* activity
  - Number of concurrent users
  - Request arrival rates
  - Performance requirements
- Later, add large scenarios, critical scenarios, etc.

Example: ATM System

System Functions:

Get balance Checking Savings	Withdrawal Checking Savings Credit card
Make payment From checking From savings In envelope	Deposit Savings Checking

Scenario?

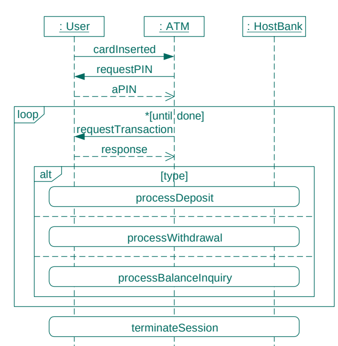
Software Specifications

- Execution paths for scenarios of interest
- Objects / methods to be executed
  - probability of execution
  - number of repetitions
  - protocol
- Database accesses, messages sent
- Level of detail increases as development progresses

Sequence Diagrams

- Sequence diagrams provide a tabular notation for
  - objects (those of interest in the scenario)
  - messages between objects
    - events
    - operation invocation
- Messages are temporally ordered
- Often created by object-oriented developers during design

ATM Sequence Diagram



Example (*continued*)

- Processing scenario: Request withdrawal
  1. Initiate session
  2. Get and interpret request
     {response = withdrawal, checking acct}
  3. Trans Authorize (Withdrawal)
  4. Dispense cash
  5. Print receipt
  6. Terminate session
- Performance requirement: Response time ________ secs.
- Workload intensity, e.g., number of session arrivals per hr. per ATM

Environment

- Platform and network characteristics:
  - configuration
  - device service rates
- Software overhead, e.g., database path lengths, communication overhead, etc.
- External factors, e.g., peak hours, bulk arrivals, batch windows, scheduling policies
- Case study Environment Specifications:
  - time for ATM communication
  - CPU speed
  - system configuration (devices, service times, etc.)

Resource Usage

Estimates and Bounds

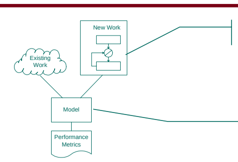
- CPU
  - Work Units
  - or # of instructions executed
  - or measurements of similar software
- I/O
- Database calls
- Communication
- Other devices
- Software overhead calls & characteristics

Example Specifications

Component	K Instr	Disk I/O's	ATM Screens
initiate session	200	1	2
get & interpret request	150	0	2
withdrawal	250	1	0
dispense cash	350	1	1
print balance	400	0	1
terminate session	100	0	1

Part 3: SPE Model-based Approach

System Versus Software Modeling Tools



Software Performance Model

Execution Graphs (may be spread-sheet, pseudo-code, etc. views of EGs)

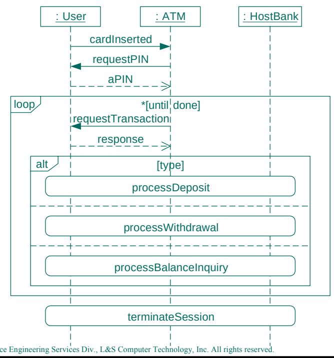
System Execution Model

Queueing Networks

System	Software
Requires more modeling expertise	Requires less modeling expertise
Device usage, overall response time and throughput	Time and resource requirements of processing steps and overall
Useful to evaluate hardware changes	Useful to evaluate software alternatives

A combination is best.

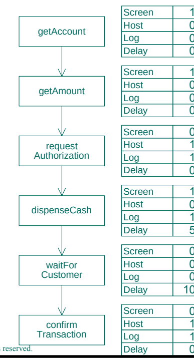
1. Start with Sequence Diagram



 31

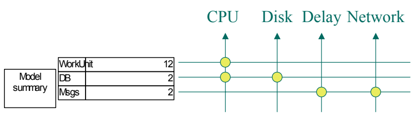
2. Software Performance Models

- Create from sequence diagrams
- Specify software resource requirements
- Specify computer resource requirements (e.g., path lengths)
- Results point to problem areas



 32

3. Computer Resource Requirements



- Ties software resource requirements to devices and service required in computer environment

 33

Overhead Matrix: Computer Resource Requirements

Devices	CPU	Disk	Display	Delay	Net
Devices	1	1	1	1	1
Service Units	Sec.	Phys. I/O	Screens	Units	Msgs.
Screen	0.001		1		
Host	0.005			3	2
Log	0.001	1			
Delay				1	
Service Time	1	0.02	1	1	0.05

 34

Software Models: Purpose

- Derive software execution characteristics
- Initial check of performance requirements
- Provide workload parameters for the system execution model

 35

Execution Graphs

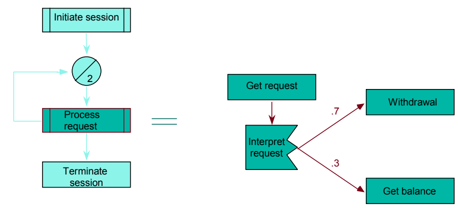
- Visual representation of software
  - Model confirmation
  - Presentation of results
- Formal basis for analysis
  - Aids in comprehension
  - Specification for tool development

 36

Model Notation

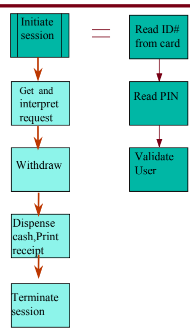
Basic Node	Processing step at current level of detail.
Expanded Node	The step has more details. Details are in the associated subgraph.
Repetition Node	The node in the loop is repeated *n* times. It may be an expanded node.
Case Node	Attached nodes are conditionally executed. A probability is associated with each.

Typical Structures

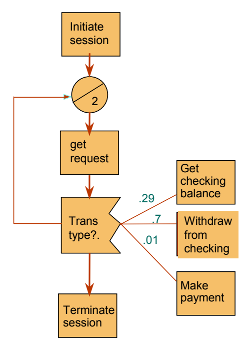


Example: Specific scenario

Withdrawal scenario:



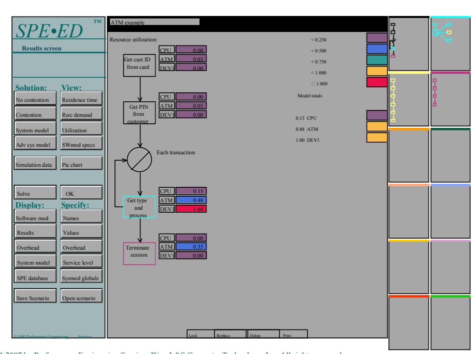
Example: Generic Scenario



Graph Analysis

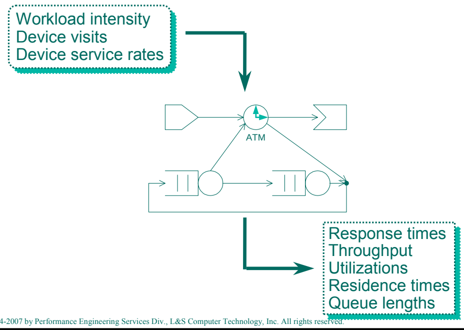
- Best case
- Worst case
- Average
- Variance
- Distribution

4. Software Model Results



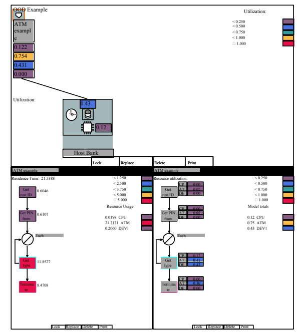
5. System Performance Model

System execution model quantifies contention effects



 43

6. System Performance Model Results



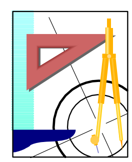
 44

Part 3: Case Study

 45

ICAD

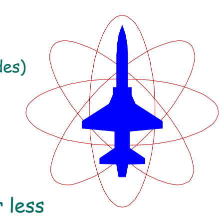
- Computer-Aided Design (CAD) application
- Features
  - interactively construct, view, and analyze drawings that model structures (e.g., aircraft wings)
  - model stored in database
  - several versions of a model may exist in the database



 46

ICAD Model

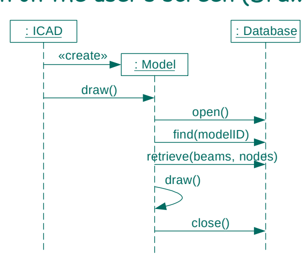
- Each models consists of
  - nodes
    - defined by position (x, y, z)
    - contain additional information needed for model solution
  - elements
    - beams (connect two nodes)
    - triangles (connect three nodes)
    - plates (connect four or more nodes)
- Typical model
  - contains only nodes and beams
  - consists of 2050 beams
- Performance requirement
  - draw typical model in 10 sec or less



 47

DrawMod Scenario

- Focus on the scenario in which a typical model is drawn on the user's screen (DrawMod)

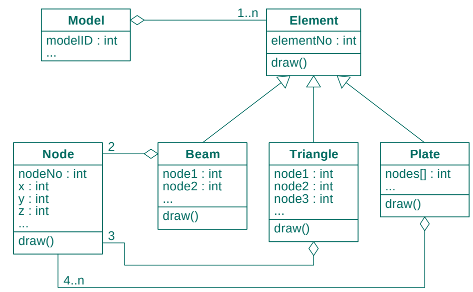


 48

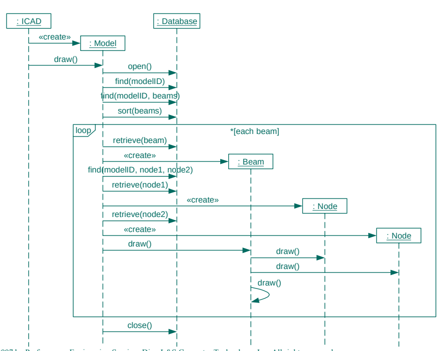
Architecture 1

- Uses an object for each node and beam

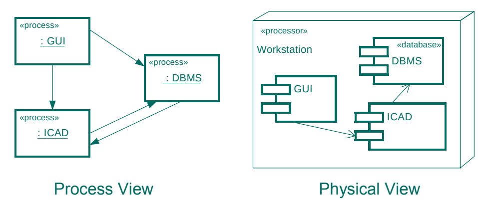
Architecture 1 – Class Diagram



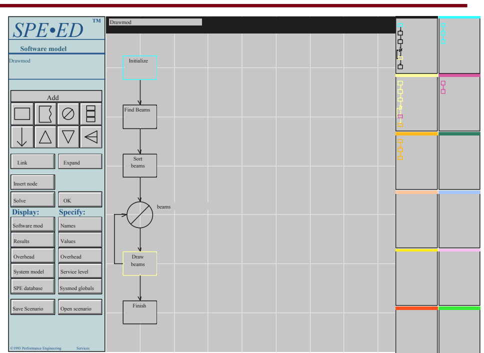
Architecture 1 – Drawmod Scenario



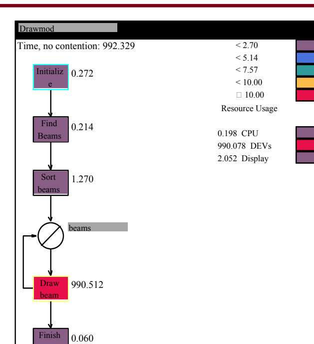
Process and Physical Views



Architecture 1 – Execution Graph



Architecture 1– Performance Results



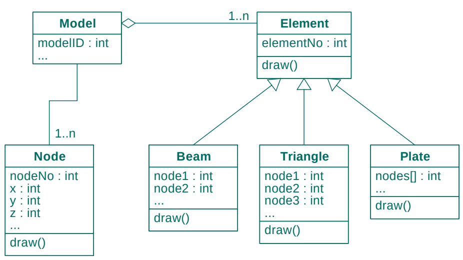
Performance Results

Architecture	Overall Response Time
Architecture 1	992.33 sec

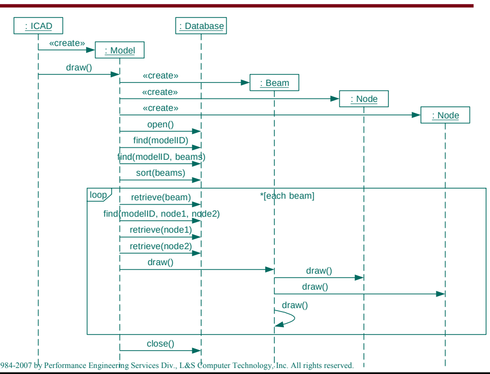
Architecture 2

- Uses "Flyweight" pattern
  - uses a shared object for beams and nodes rather than individual objects
  - reduces run-time overhead due to constructor invocations

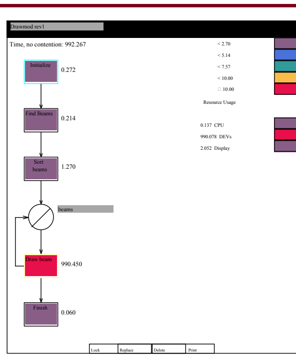
Architecture 2 – Class Diagram



Architecture 2 – Drawmod Scenario



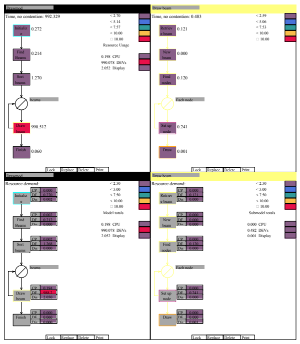
Architecture 2 – Performance Results



Performance Results

Architecture	Overall Response Time
Architecture 1	992.33 sec
Architecture 2	992.27 sec

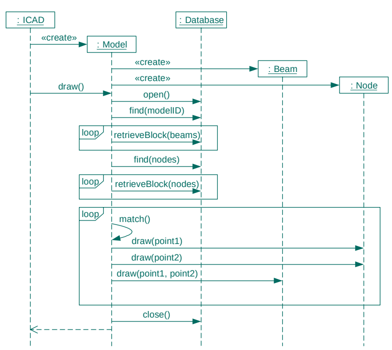
Architecture 1– Performance Results



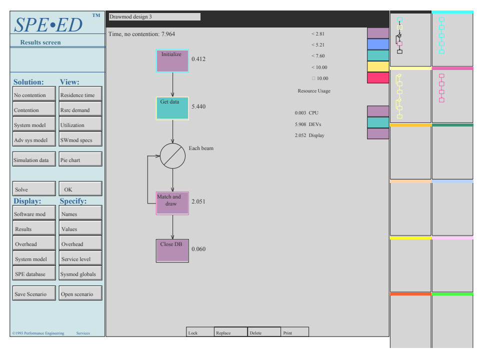
Architecture 3

- Retains use of “Flyweight” pattern
- Modifies database management system
  - new operation – retrieveBlock()
  - retrieves block of data rather than individual nodes and beams
  - single block retrieve get 20k
    - 64 beams, or
    - 170 nodes
  - reduces database accesses to
    - 33 to retrieve beams, plus
    - 9 to retrieve nodes

Architecture 3 – Drawmod Scenario



Architecture 3 – Execution Graph



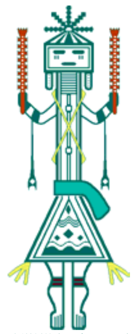
Performance Results

Architecture	Overall Response Time
Architecture 1	992.33 sec
Architecture 2	992.27 sec
Architecture 3	~8 sec

Beyond Simple Models

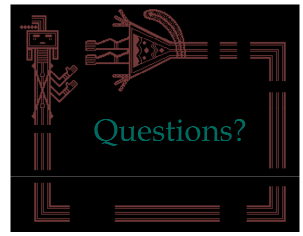
- SPE: guidance for conducting steps in process
  - e.g. performance walkthroughs, techniques for conducting studies, etc.
- Advanced modeling techniques
  - Distributed and web-based system models
  - Real-time, embedded computer systems
- Principles, Patterns, and Antipatterns
  - Techniques for performance-oriented design
- Tools for SPE

Summary



- Significance of early assessment of software performance
- Software Performance Engineering (SPE) Review
- Case Study: Use of SPE to perform early assessments of software architectures
- Additional facets of SPE

67



cusmith@spe-ed.com
http://www.spe-ed.com/

68