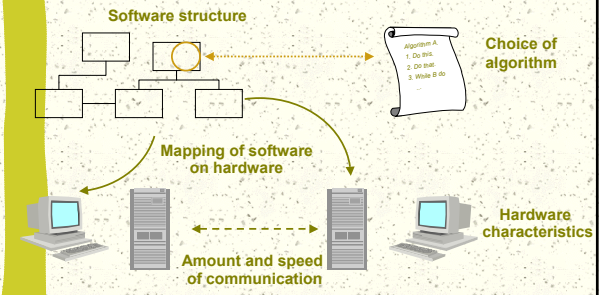
Software performance

Inkeri Verkamo

Performance prediction
Software performance models
Case example

What affects performance?



Any of these can be influenced if addressed early enough!

Uses of performance modeling

- Performance prediction
  - *what will be the reponse time of request Q on the future system?*
- Capacity analysis
  - *how many components X, Y, and Z do we need for a workload of N requests/second?*
- Sensitivity analysis
  - *if we replace component Y by Y', how will the response time of request Q change?*

If this is not acceptable, improve (before its too late)!

These can be hardware or software components.

The times can be either predicted or measured.

Performance evaluation vs. prediction

Evaluation:

- measuring the behavior of an implemented system
- actual performance numbers
- if performance is inadequate, only few cheap and easy ways to improve

Prediction:

- predicting the behavior of a future system
- estimates (may be not very accurate)
- more flexibility for improvement
- suggestions on what should be improved

Performance prediction

- Place your improvement effort right:
  - find out which parts will be crucial with respect to performance
- Early phase of development:
  - nothing implemented = nothing to measure
  - only rough estimates from previous work, prototypes, …
  - simple models for quick estimates
  - evaluate the sensitivity of your results for inaccuracy of input

Software performance modeling

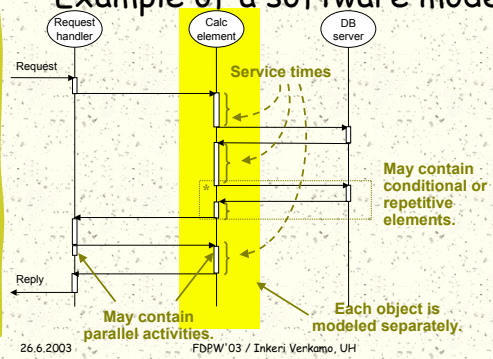
1. Define the system as a collection of objects providing services to each other
2. Model the behavior of each object separately
   ⇒ (set of) software performance models
3. Model the cooperation of the objects:
   - parallelism (resource contention)
   - synchronisation

   ⇒ system performance model

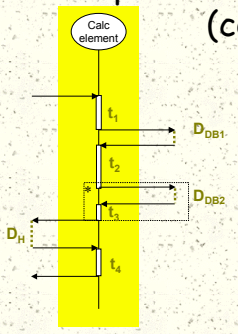
Software performance models

- Possible representations:
  - UML behavioral diagrams (e.g. sequence diagrams or activity diagrams)
  - algorithms
  - special diagrams such as execution graphs
- Level of detail:
  - what is needed to describe essentials of resource usage (computation, database services, messaging, ...)
  - repetition, conditionality, parallelism, if they affect resource usage
- A rough estimate for resource usage

Example of a software model



Example of a software model (cont.)



Resource usage of this object (for each type of resource):

$T = t_1 + t_2 + n*t_3 + t_4$

Time effect of other objects is represented as delays.

Total elapsed time (best case):

$T + D_H + D_{DB1} + n*D_{DB2}$

System performance models

- Use the results of the software models as input to the system model
- Possible modeling techniques are, e.g., queueing networks and Petri nets
- Study the effect of many requests using the resources at the same time:
  - synchronisation
  - resource contention
- Capacity needs ⇔ configuration issues

What input do we need?

- Key performance scenarios
- Workload for each key scenario
- Structure of the system
- Load caused by each scenario on each element
- Performance requirements with respect to the scenarios

We need some numbers for each item

... but we can live with inaccuracy!

Sources of input data

- System requirements documentation:
  - key scenarios
  - projected workload
  - performance requirements
- System architecture description:
  - system structure
- Test runs: load caused by scenarios
- Previous system measurements:
  - reality checks for all types of input data

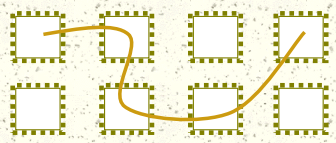
No one knows the exact numbers, but many know some useful ones!

Case example: telephone switch

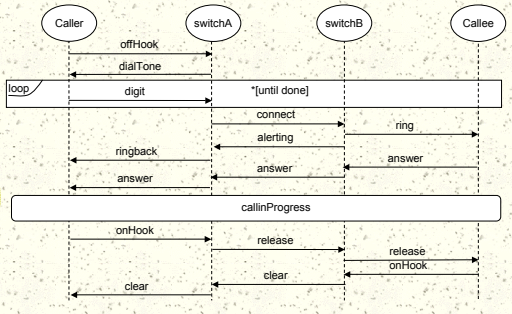
- soft real-time system
- as many calls per hour as possible
- central use cases:
  - placing a phone call
  - receiving a phone call

Architecture

- modular structure:
  - each module handles a group of lines
  - switch is scalable by adding more modules
  - for each phone call a path is established from the caller module to the callee module



Call sequence diagram



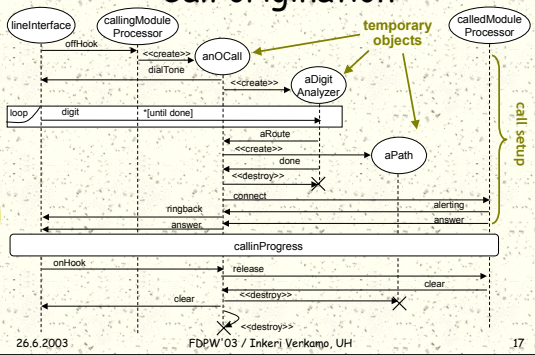
Performance scenarios

- separate scenario for each phase:
  - call origination
  - call termination
  - call in progress
    - duration etc. varies a lot
  - caller hang up
  - callee hang up

most interesting scenario:

how long must the caller wait, before the call is established?

Call origination



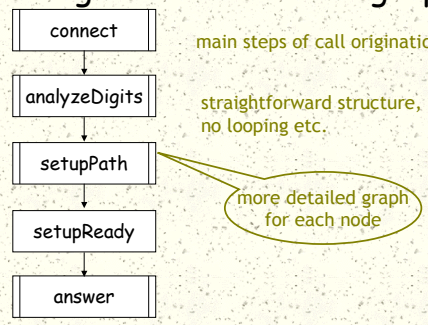
Software resources

- central software resources:
  - processor
    - unit: KInstructions
    - service time (depends on hardware): 0.000015 s
  - line interface:
    - processing and message passing
    - unit: number of messages = number of visits to the line interface
    - each visit to the line interface requires
      - processing: 100 KInstr
      - message passing delay: 0.005 s

Performance goals

- workload:
  - 3 call originations / s
  - same number of call terminations
  - assumption:
    - half of the hang-ups by caller, half by callee
- call setup time:
  - critical from point of view of the switch
  - on the average at most **0.5 s**

Call origination execution graph



Resource usage

	CPU	Line I/F	Time
connect	3400 KInstr	1 visit	0.056 s
analyzeDigits	2400 KInstr	0 visits	0.036 s
setupPath	4350 KInstr	5 visits	0.090 s
setupReady	550 KInstr	1 visit	0.013 s
answer	1000 KInstr	2 visits	0.025 s
	11700 Kinstr	9 visits	**0.220 s**

setup time

best case time < 0.5 s

System model

- system model with one scenario (call origination):
  - 3 requests / s
  - ⇒ setup time: **0.38 s**
- system model with all four scenarios:
  - 3 requests / s for each scenario
  - simulation result: setup time **>16 s !**
  - explanation: CPU saturates

System improvement

- reduce CPU use:
  - save time spent in creating and destroying objects by using recycled call objects
  - corresponding change in all scenarios
- simulation results:
  - CPU utilization **68%**
  - setup time **0.31 s**

Fixing the problem would have been much more expensive, if it had been found after implementation.

Literature

- C.U. Smith, L.G. Williams, *Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software;* Addison-Wesley 2001
- Workshop on Software and Performance (WOSP)
  - 1998, 2000, 2002
  - next workshop in January 2004