Model Interchange Formats: PMIF, S-PMIF, Supporting Tools

Objectives

- Motivation
- S-PMIF
- PMIF Core & Prototypes
- Building on PMIF

Part 1: Motivation

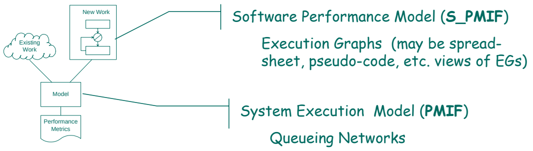
Motivation for Tool Interoperability

- Gap between software developers and performance specialists
- Economics/expertise required for building "tool for everything"
- Tools should specialize in what they do best and share knowledge with other tools

Our Research Strategy

- Bridge a variety of design and modeling tools
- Re-use existing tools when appropriate
- De-skill the performance modeling & performance decision support -> empower developers who need performance info

System Versus Software Modeling Tools



Software Performance Model (**S_PMIF**)

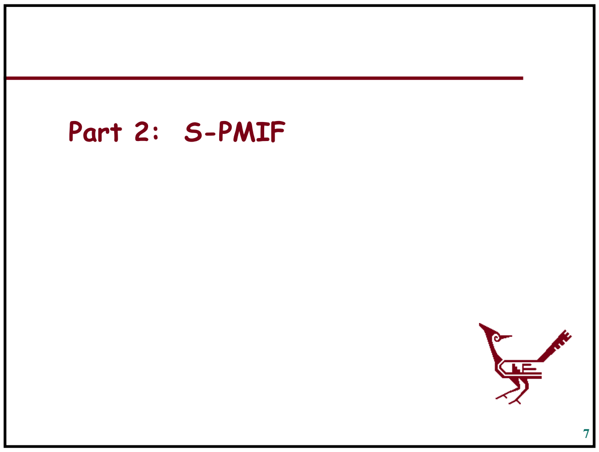
Execution Graphs (may be spread-sheet, pseudo-code, etc. views of EGs)

System Execution Model (**PMIF**)

Queueing Networks

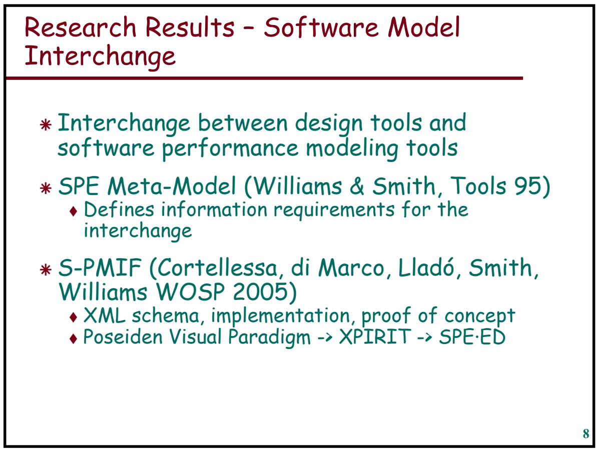
System	**Software**
Requires more modeling expertise	Requires less modeling expertise
Device usage, overall response time and throughput	Time and resource requirements of processing steps and overall
Useful to evaluate hardware changes	Useful to evaluate software alternatives

A combination is best.



Part 2: S-PMIF

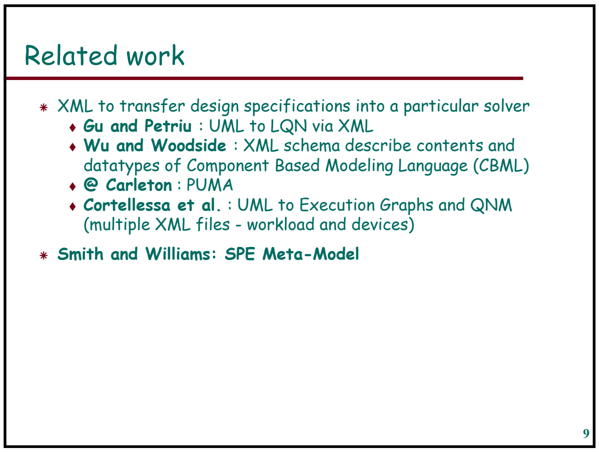
7



Research Results – Software Model Interchange

- Interchange between design tools and software performance modeling tools
- SPE Meta-Model (Williams & Smith, Tools 95)
  - Defines information requirements for the interchange
- S-PMIF (Cortellessa, di Marco, Lladó, Smith, Williams WOSP 2005)
  - XML schema, implementation, proof of concept
  - Poseiden Visual Paradigm -> XPIRIT -> SPE·ED

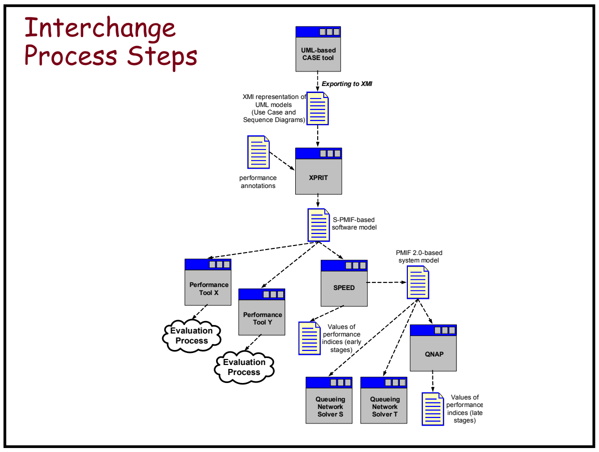
8



Related work

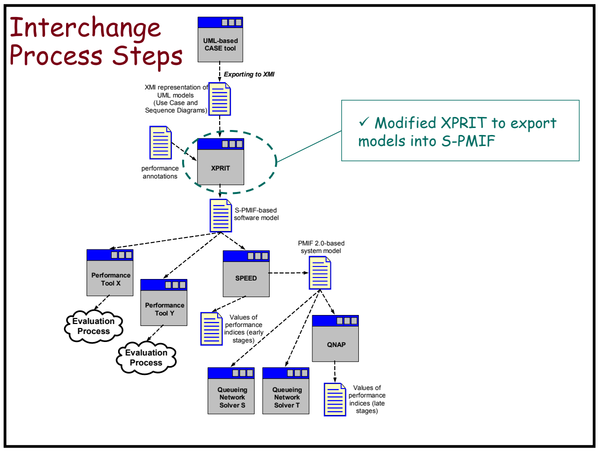
- XML to transfer design specifications into a particular solver
  - **Gu and Petriu** : UML to LQN via XML
  - **Wu and Woodside** : XML schema describe contents and datatypes of Component Based Modeling Language (CBML)
  - **@ Carleton** : PUMA
  - **Cortellessa et al.** : UML to Execution Graphs and QNM (multiple XML files - workload and devices)
- **Smith and Williams: SPE Meta-Model**

9



Interchange Process Steps

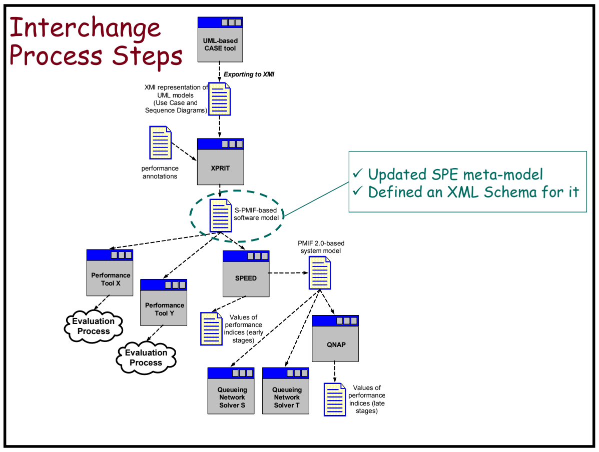
UML-based CASE tool
Exporting to XMI
XMI representation of UML models (Use Case and Sequence Diagrams)
performance annotations
XPRIT
S-PMIF-based software model
PMIF 2.0-based system model
Performance Tool X
Performance Tool Y
SPEED
Evaluation Process
Evaluation Process
Values of performance indices (early stages)
QNAP
Queueing Network Solver S
Queueing Network Solver T
Values of performance indices (late stages)



Interchange Process Steps

UML-based CASE tool
Exporting to XMI
XMI representation of UML models (Use Case and Sequence Diagrams)
performance annotations
XPRIT
S-PMIF-based software model
PMIF 2.0-based system model
Performance Tool X
Performance Tool Y
SPEED
Evaluation Process
Evaluation Process
Values of performance indices (early stages)
QNAP
Queueing Network Solver S
Queueing Network Solver T
Values of performance indices (late stages)

✓ Modified XPRIT to export models into S-PMIF

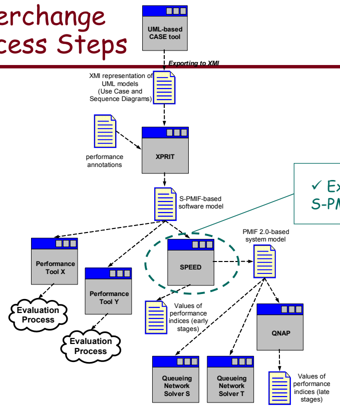


Interchange Process Steps

UML-based CASE tool
Exporting to XMI
XMI representation of UML models (Use Case and Sequence Diagrams)
performance annotations
XPRIT
S-PMIF-based software model
PMIF 2.0-based system model
Performance Tool X
Performance Tool Y
SPEED
Evaluation Process
Evaluation Process
Values of performance indices (early stages)
QNAP
Queueing Network Solver S
Queueing Network Solver T
Values of performance indices (late stages)

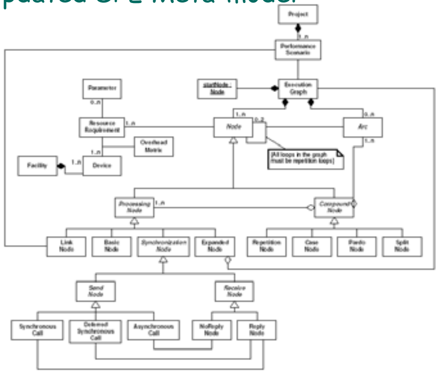
✓ Updated SPE meta-model
✓ Defined an XML Schema for it

Interchange Process Steps


UML-based CASE tool
Exporting to XMI
XMI representation of UML models (Use Case and Sequence Diagrams)
performance annotations
XPRIT
S-PMIF-based software model
Performance Tool X
Performance Tool Y
Evaluation Process
Evaluation Process
SPEED
PMIF 2.0-based system model
Values of performance indices (early stages)
QNAP
Queueing Network Solver S
Queueing Network Solver T
Values of performance indices (late stages)

✓ Extended SPE·ED to import S-PMIF models

Updated SPE Meta-model

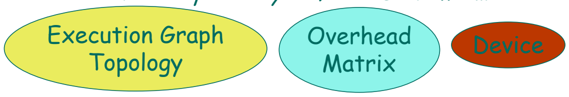


Updated SPE meta-model
- *Most relevant updates* -

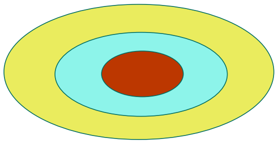
- Deleted *State Identification Node*
- Added *Synchronization Node* and its sub-classes
- Added *Facility*
- Added *Project* with multiple scenarios
- Modified the *Device* definition to better specify characteristics of different types of devices (e.g. CPU, Disk, etc.)
- Other minor changes

Defining an XML Schema for the SPE meta-model

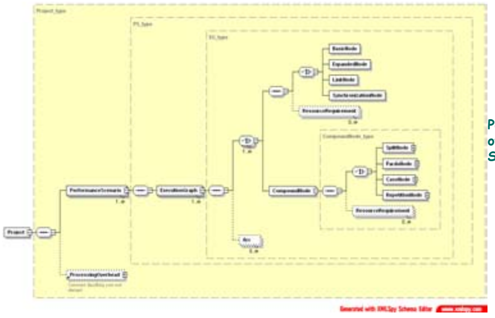
<u>High level</u> structure (separation of concerns) **:** *three separately defined Schemas…*



… that, upon assembled, represent the SPE meta-model



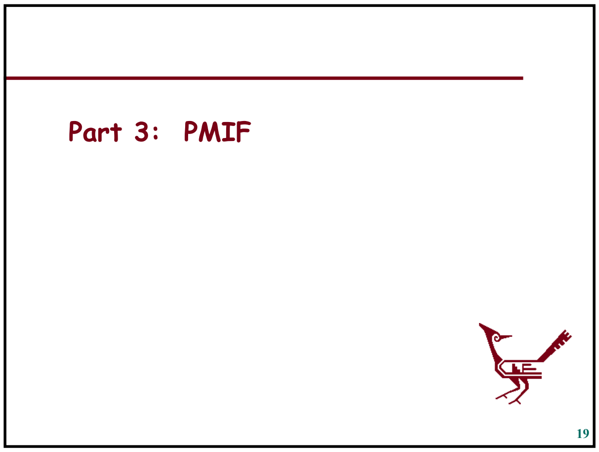
XML Schema for the SPE meta-model



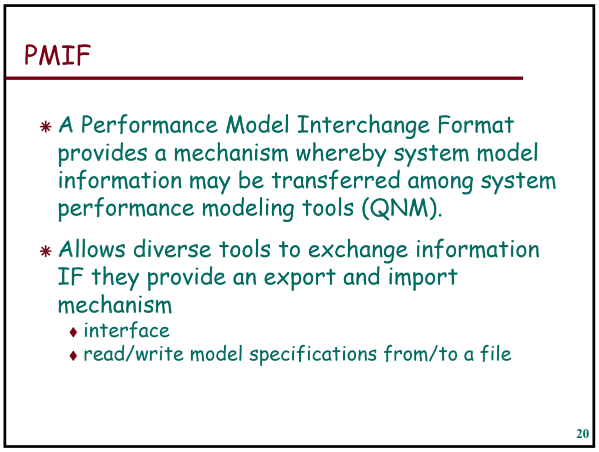
Portion of the Schema

Results

- Proof of Concept (WOSP05) ++
  - Extended XPRIT to export models into S-PMIF
  - Extended SPE·ED to import S-PMIF models
  - From SPE·ED to QNAP using PMIF
  - Experimental results
  - UML 2.0
- Observations
  - Sometimes UML model **semantics are not unique**, so (apparently) **redundant info** is needed to interpret and translate the model
  - Automatically generated performance models may suffer from an **analogous non-optimization as automatically generated code**
  - Resulting models are limited by incoming data. Model schematics alone are not useful.

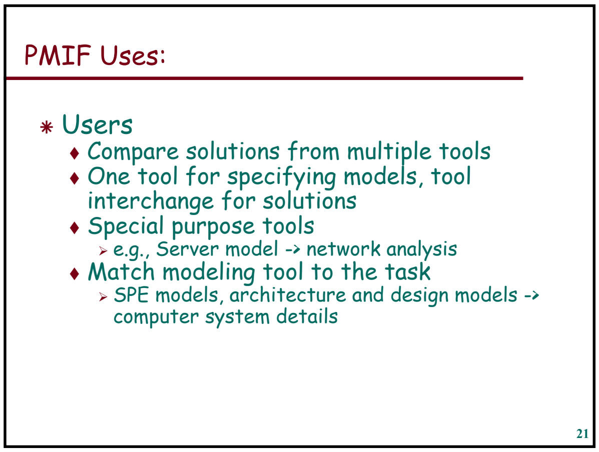


Part 3: PMIF



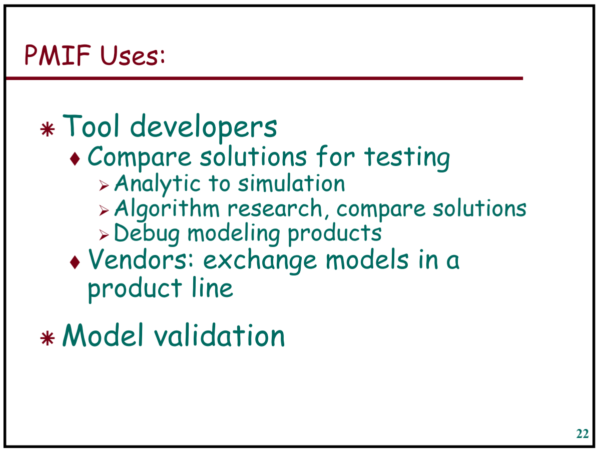
PMIF

- A Performance Model Interchange Format provides a mechanism whereby system model information may be transferred among system performance modeling tools (QNM).
- Allows diverse tools to exchange information IF they provide an export and import mechanism
  - interface
  - read/write model specifications from/to a file



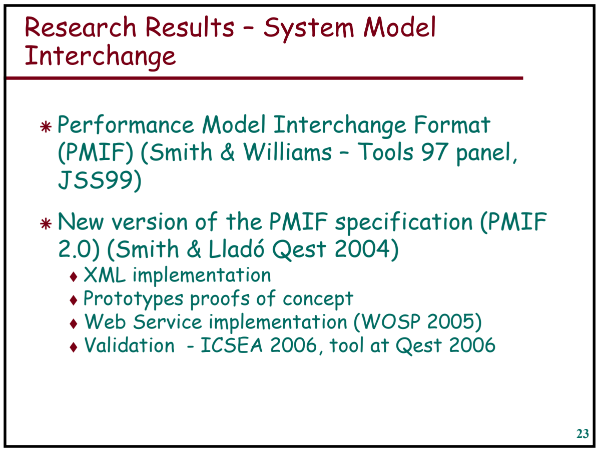
PMIF Uses:

- Users
  - Compare solutions from multiple tools
  - One tool for specifying models, tool interchange for solutions
  - Special purpose tools
    - e.g., Server model -> network analysis
  - Match modeling tool to the task
    - SPE models, architecture and design models -> computer system details



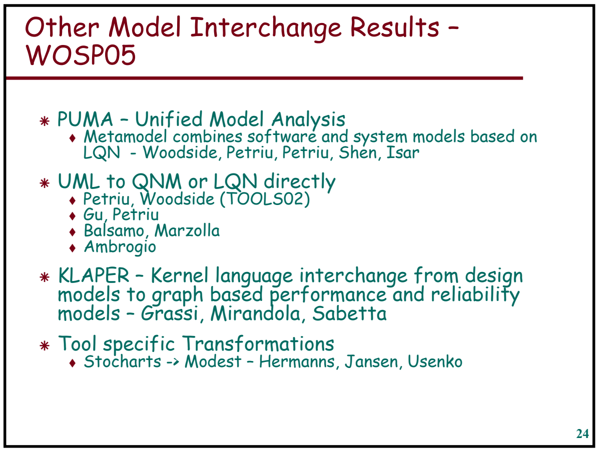
PMIF Uses:

- Tool developers
  - Compare solutions for testing
    - Analytic to simulation
    - Algorithm research, compare solutions
    - Debug modeling products
  - Vendors: exchange models in a product line
- Model validation



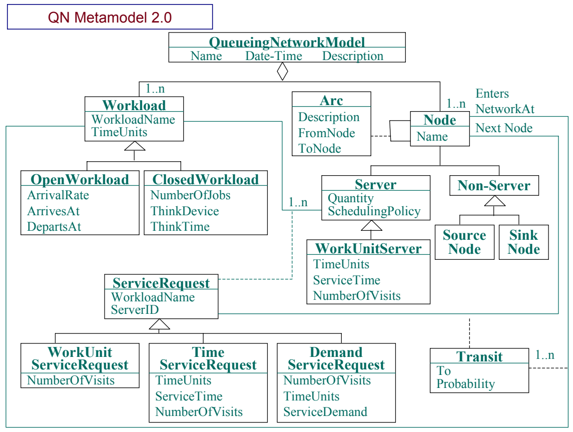
Research Results – System Model Interchange

- Performance Model Interchange Format (PMIF) (Smith & Williams – Tools 97 panel, JSS99)
- New version of the PMIF specification (PMIF 2.0) (Smith & Lladó Qest 2004)
  - XML implementation
  - Prototypes proofs of concept
  - Web Service implementation (WOSP 2005)
  - Validation - ICSEA 2006, tool at Qest 2006

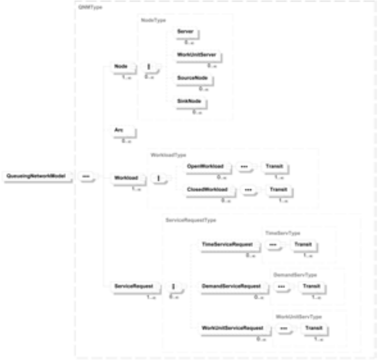


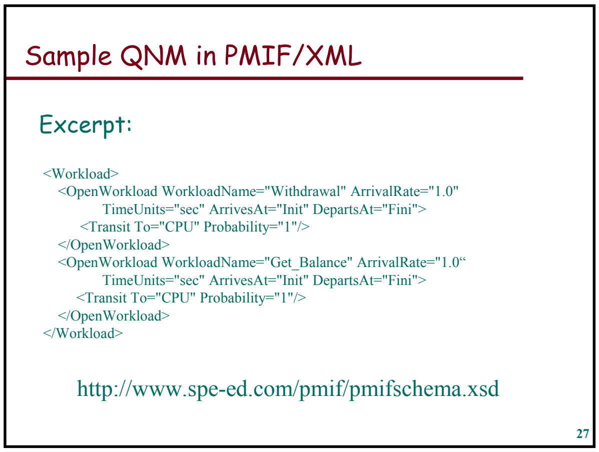
Other Model Interchange Results – WOSP05

- PUMA – Unified Model Analysis
  - Metamodel combines software and system models based on LQN - Woodside, Petriu, Petriu, Shen, Isar
- UML to QNM or LQN directly
  - Petriu, Woodside (TOOLS02)
  - Gu, Petriu
  - Balsamo, Marzolla
  - Ambrogio
- KLAPER – Kernel language interchange from design models to graph based performance and reliability models – Grassi, Mirandola, Sabetta
- Tool specific Transformations
  - Stocharts -> Modest – Hermanns, Jansen, Usenko



PMIF 2.0 XML Schema



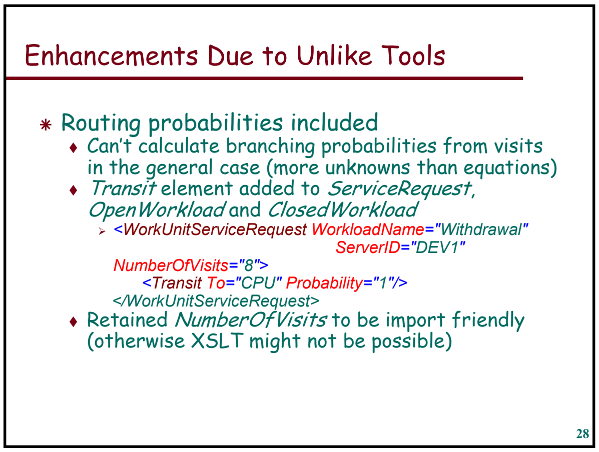


Sample QNM in PMIF/XML

Excerpt:

```
<Workload>
   <OpenWorkload WorkloadName="Withdrawal" ArrivalRate="1.0"
            TimeUnits="sec" ArrivesAt="Init" DepartsAt="Fini">
       <Transit To="CPU" Probability="1"/>
   </OpenWorkload>
   <OpenWorkload WorkloadName="Get_Balance" ArrivalRate="1.0“
            TimeUnits="sec" ArrivesAt="Init" DepartsAt="Fini">
       <Transit To="CPU" Probability="1"/>
   </OpenWorkload>
</Workload>
```

http://www.spe-ed.com/pmif/pmifschema.xsd

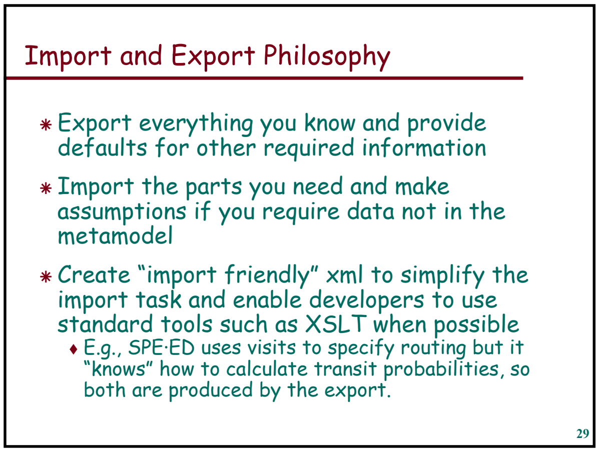


Enhancements Due to Unlike Tools

- Routing probabilities included
  - Can't calculate branching probabilities from visits in the general case (more unknowns than equations)
  - *Transit* element added to *ServiceRequest*, *OpenWorkload* and *ClosedWorkload*

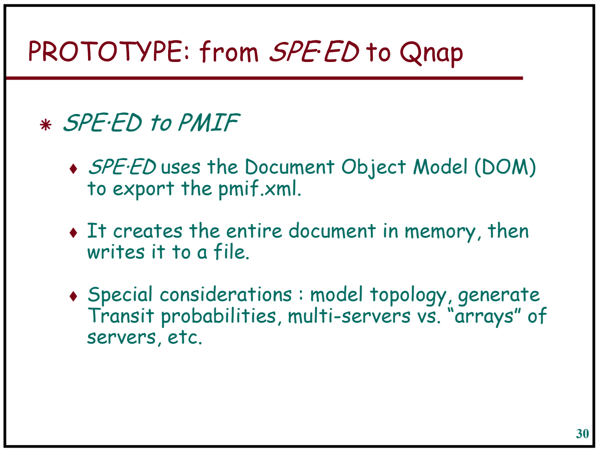
```
<WorkUnitServiceRequest WorkloadName="Withdrawal"
                        ServerID="DEV1"
NumberOfVisits="8">
     <Transit To="CPU" Probability="1"/>
</WorkUnitServiceRequest>
```

  - Retained *NumberOfVisits* to be import friendly (otherwise XSLT might not be possible)



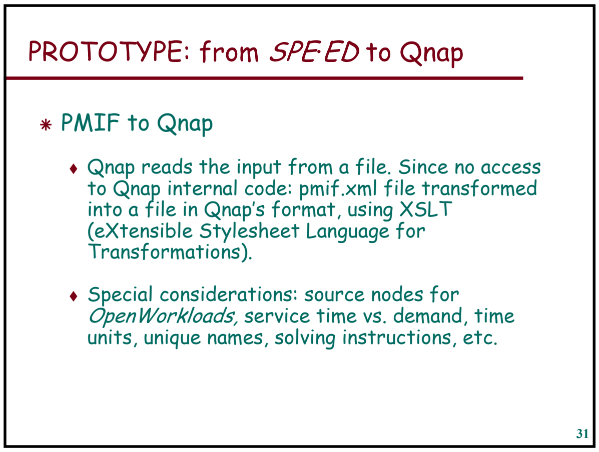
Import and Export Philosophy

- Export everything you know and provide defaults for other required information
- Import the parts you need and make assumptions if you require data not in the metamodel
- Create “import friendly” xml to simplify the import task and enable developers to use standard tools such as XSLT when possible
  - E.g., SPE·ED uses visits to specify routing but it “knows” how to calculate transit probabilities, so both are produced by the export.



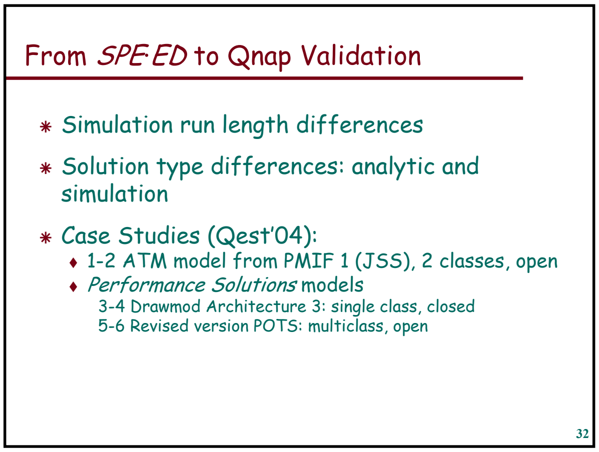
PROTOTYPE: from *SPE·ED* to Qnap

- *SPE·ED to PMIF*
  - *SPE·ED* uses the Document Object Model (DOM) to export the pmif.xml.
  - It creates the entire document in memory, then writes it to a file.
  - Special considerations : model topology, generate Transit probabilities, multi-servers vs. “arrays” of servers, etc.



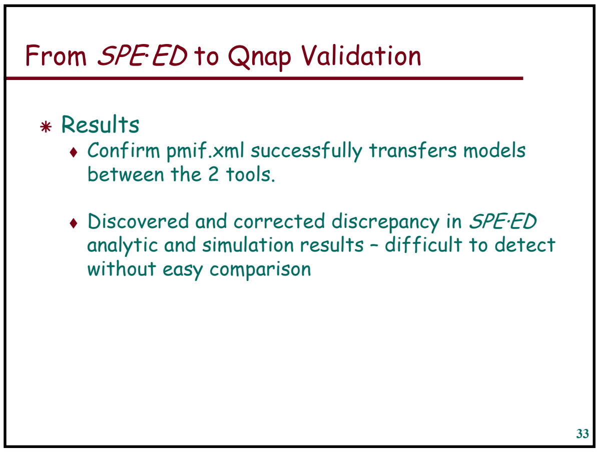
PROTOTYPE: from *SPE·ED* to Qnap

- PMIF to Qnap
  - Qnap reads the input from a file. Since no access to Qnap internal code: pmif.xml file transformed into a file in Qnap's format, using XSLT (eXtensible Stylesheet Language for Transformations).
  - Special considerations: source nodes for *OpenWorkloads*, service time vs. demand, time units, unique names, solving instructions, etc.



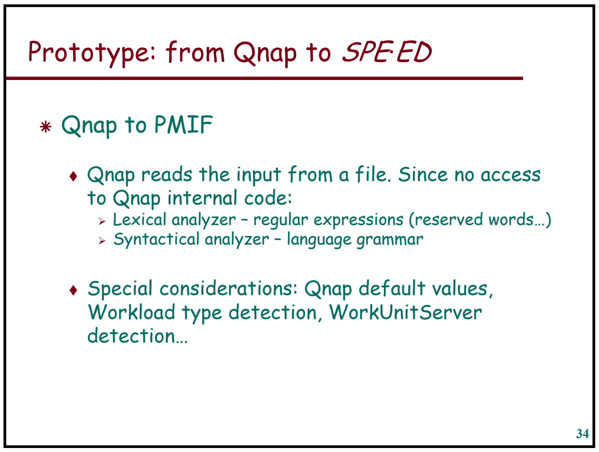
From *SPE·ED* to Qnap Validation

- Simulation run length differences
- Solution type differences: analytic and simulation
- Case Studies (Qest'04):
  - 1-2 ATM model from PMIF 1 (JSS), 2 classes, open
  - *Performance Solutions* models
    - 3-4 Drawmod Architecture 3: single class, closed
    - 5-6 Revised version POTS: multiclass, open



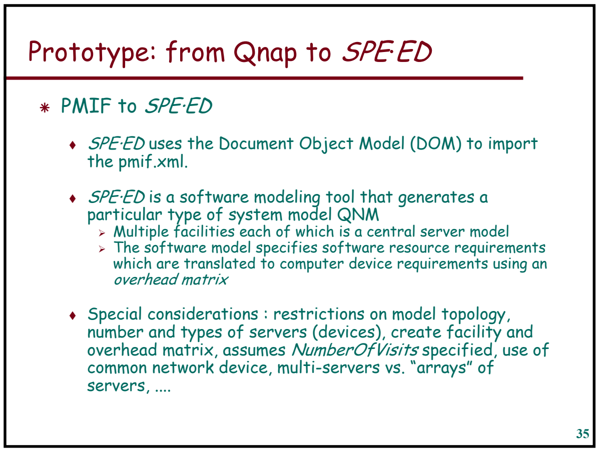
From *SPE·ED* to Qnap Validation

- Results
  - Confirm pmif.xml successfully transfers models between the 2 tools.
  - Discovered and corrected discrepancy in *SPE·ED* analytic and simulation results – difficult to detect without easy comparison



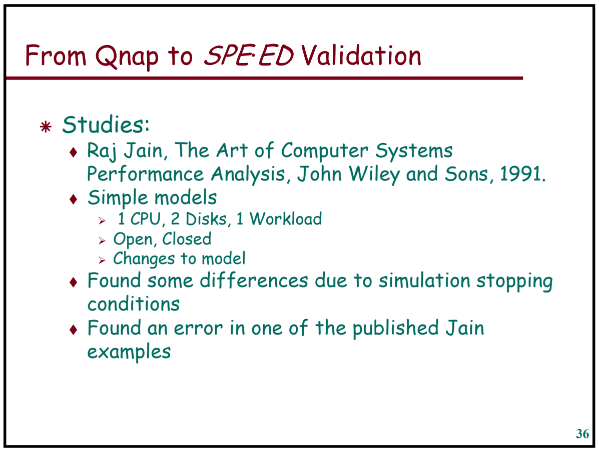
Prototype: from Qnap to *SPE·ED*

- Qnap to PMIF
  - Qnap reads the input from a file. Since no access to Qnap internal code:
    - Lexical analyzer – regular expressions (reserved words…)
    - Syntactical analyzer – language grammar
  - Special considerations: Qnap default values, Workload type detection, WorkUnitServer detection…



Prototype: from Qnap to *SPE·ED*

- PMIF to *SPE·ED*
  - *SPE·ED* uses the Document Object Model (DOM) to import the pmif.xml.
  - *SPE·ED* is a software modeling tool that generates a particular type of system model QNM
    - Multiple facilities each of which is a central server model
    - The software model specifies software resource requirements which are translated to computer device requirements using an *overhead matrix*
  - Special considerations : restrictions on model topology, number and types of servers (devices), create facility and overhead matrix, assumes *NumberOfVisits* specified, use of common network device, multi-servers vs. "arrays" of servers, ….



From Qnap to *SPE·ED* Validation

- Studies:
  - Raj Jain, The Art of Computer Systems Performance Analysis, John Wiley and Sons, 1991.
  - Simple models
    - 1 CPU, 2 Disks, 1 Workload
    - Open, Closed
    - Changes to model
  - Found some differences due to simulation stopping conditions
  - Found an error in one of the published Jain examples

Jain '91 example: Validation Results

	Book (Jain))	Qnap			SPE·ED	
	Convolution	MVA	Sim 100000	Sim 1000000	MVA	Sim 100000
Response Time	6,6940	6,6940	6,6940	6,7000	6,6680	6,6942
Throughput	0,2640	0,2638	0,2638	0,2628	0,2644	0,2638
CPU Residence	0,0450	0,0454	0,0454	0,0454	0,0454	0,0454
DiskA Residence	0,2830	0,2825	0,2825	0,2836	0,2819	0,2825
DiskB Residence	0,3520	0,3522	0,3522	0,3532	0,3518	0,3522
CPU Utilization	0,2060	0,2057	0,2057	0,2049	0,2056	0,2100
DiskA Utilization	0,6180	0,6172	0,6172	0,6179	0,6162	0,6200
DiskB Utilization	0,4120	0,4115	0,4115	0,4092	0,4118	0,4100

Conclusions

- PMIF enables the interchange of system model information based on QNM
- Proof of concept using unlike tools demonstrates the viability
  - Comparison of tool results across tools is beneficial
- Importing and exporting tools can implement the functions internally, or file transformations may be used without requiring tool developers to modify code

Part 3: Building on PMIF

PMIF Semantic Validation: Motivation

```
<QueueingNetworkModel xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
Name="Test">
    <Node>
        <SourceNode Name="SourceNode"/>
        <Server Name="CPU" Quantity="1" SchedulingPolicy="PS"/>
        <SinkNode Name="SinkNode“/>
    </Node>
    <Arc FromNode="SourceNode" ToNode="OpenWL"/>
    <Arc FromNode="CPU" ToNode="SinkNode"/>
    <Workload>
        <OpenWorkload WorkloadName="OpenWL" ArrivalRate="3"
        ArrivesAt = “SourceNode” DepartsAt="SinkNode">
            <Transit To="CPU" Probability="1"/>
        </OpenWorkload>
    </Workload>
    <ServiceRequest>
        <DemandServiceRequest WorkloadName="OpenWL" ServerID="CPU"
        ServiceDemand="0.123" TimeUnits="min" NumberOfVisits="10">
            <Transit To="SinkNode" Probability="0.5"/>
            <Transit To="CPU" Probability="0.5"/>
        </DemandServiceRequest>
    </ServiceRequest>
</QueueingNetworkModel>
```

PMIF Semantic Validation Uses

- PMIF import tools: only one validation code
- PMIF export tools: to check that they generate correct models
- Web Service, developed, installed and maintained once for all its users

XML Semantic Validation Approaches

1. Domain-specific custom programs
2. XSLT stylesheets
3. Constraint specification languages

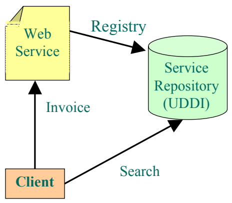
- The validations and order for checking conditions are the same regardless of approach used.
- We use a custom program

Semantic Validations

- Error Generation
  - Coherent identifiers
  - Duplicates
  - Coherent workload chains
  - Routing probability equations …
- Warnings (2 levels)
  - Elements specified but not referenced
  - Time units not specified
  - Attribute values equals zero
  - FCFS servers …

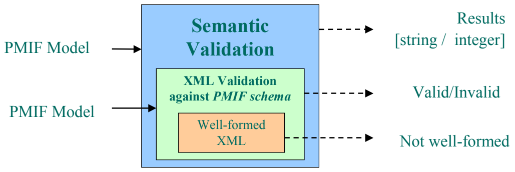
Web Services

- Web services allow communication between applications.
- Components (XML based)
  - SOAP (Simple Object Access Protocol) – works on existing transport protocols such HTTP
  - WSDL (Web Services Description Language) – methods and parameters description
  - UDDI (Universal Description, Discovery, and Integration)



Web Service Implementation

- Users that do not have Java or do not want to install and keep updated the semantic validation tool…
- Need only to have a SOAP client (any technology)

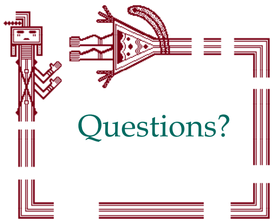


PMIF Related Extensions

- Experiments
- Results
- Simulation…

Summary

- Motivation
- S-PMIF
- PMIF Core & Prototypes
- Building on PMIF



www. spe-ed.com
dmi.uib.es/~cllado/pmif/