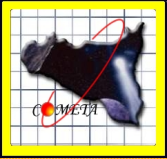


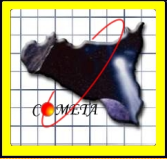
TSD topics: security, dependability and performance

*Carmelo Ragusa
University of Messina
2nd ESFORS workshop
Maribor, Slovenia
10-11/07/07*

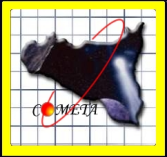


Acknowledgments

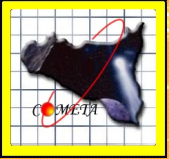
- This work makes use of results produced by the PI2S2 Project managed by the Consorzio COMETA, a project co-funded by the Italian Ministry of University and Research (MIUR) within the Piano Operativo Nazionale Ricerca Scientifica, Sviluppo Tecnologico, Alta Formazione (PON 2000-2006). More information is available at <http://www.pi2s2.it> and <http://www.consorzio-cometa.it>.



- **Topics**
 - Security
 - Dependability
 - Performance
- **Conclusions**



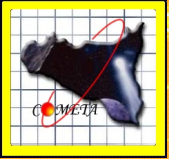
Security on Grid: *enhancing the Grid data storage service*



Issues on Grid data storage

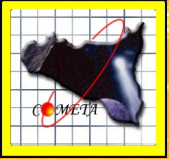
- Actual Grid middlewares provide security for user access control but no for storage --> user's data are stored in clear
- Administrators can access data on storage systems
- Enterprises are aware of the problem

→ *Data access must be guaranteed to data owners only or whoever they authorize*

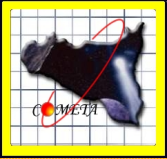


Grid Storage Security: requirements and solution

- UniMe is working to provide a solution to the above mentioned problem. The solution needs to be:
 - integrable to the existing middlewares → standards compliant
 - usable → acceptable data access
 - transparent to the user
 - secure also for communications mechanisms
- UniME solution
 - use of GFAL-like library
 - POSIX compliant
 - data symmetric encryption
 - symmetric key subdivision and secure distributed storing of the subkeys within the Grid infrastructure itself
 - implemented as a Grid service
 - use of SSL channel for communication
 - testing in the PI2S2 Grid infrastructure



Dependability evaluation of dynamic systems

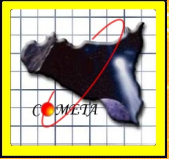


Dependability evaluation of dynamic systems

- **Main formalisms to evaluate system reliability/availability:**
 - Reliability Block Diagram (RBD)
 - Fault Tree (FT)
- **Problems and/or restrictions in modeling dynamic aspects:**
 - Stochastic independence assumption
 - Dependence, interference, redundancy, load sharing, reparability, common cause failures, ...

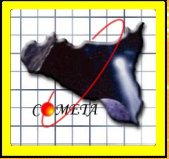
Solution: extend these formalisms with dynamic modeling capabilities

- FT → DFT (Dugan et al.)
- RBD → DRBD (Distefano et al.)

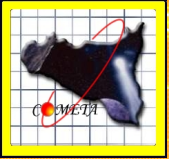


“Structured” Combinatorial Models

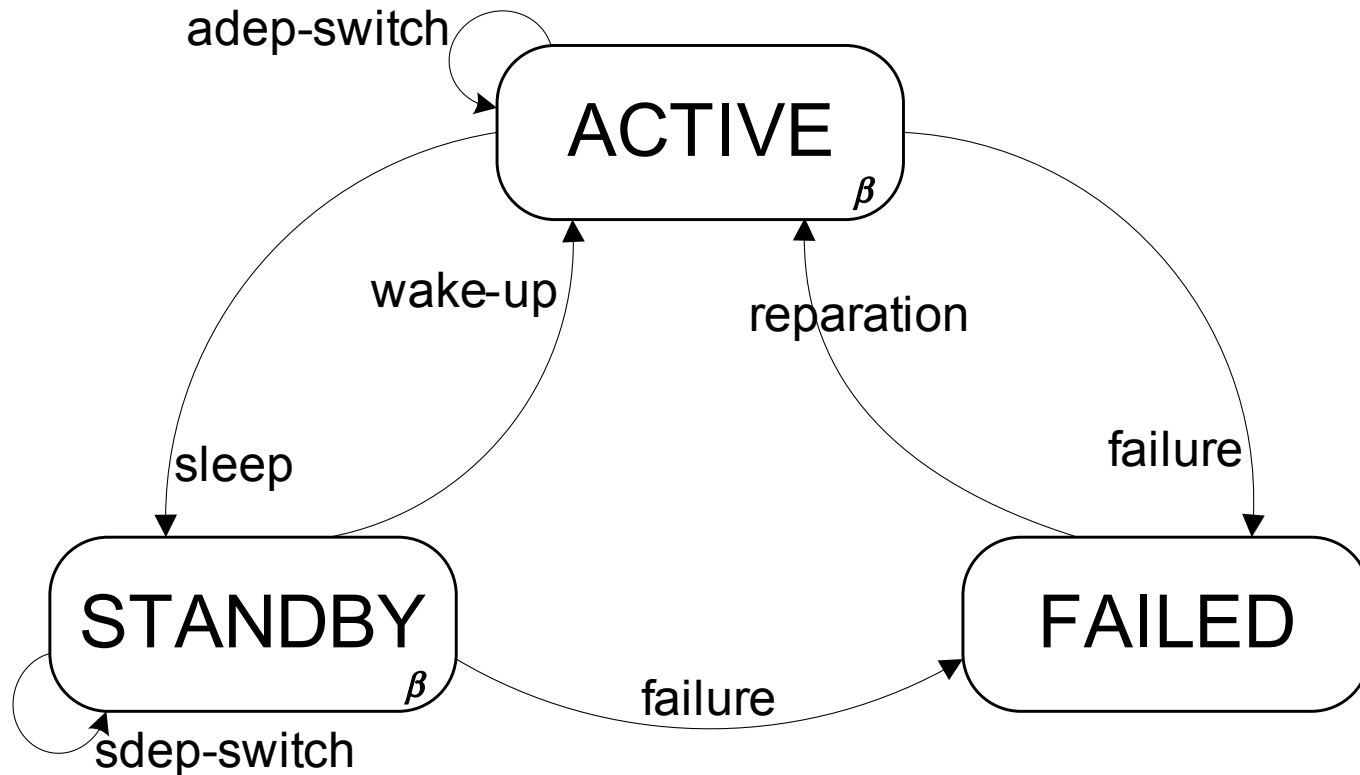
- **Fault trees and reliability block diagrams**
 - Integrate certain probability events into a module
 - Organize the modules in a “structured” way, according to the effects of each module’s failure
 - Capture conditions that make a system fail in terms of the structural relationships between the system components.
 - Assuming statistical independence
 - Failures-Repairs independence
 - Readable, easy to use and analyse (analytic computation)

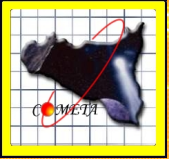


- **Extend and enhance the RBD formalism maintaining easiness in modeling and readability**
 - Dynamic model
 - States and events characterization
 - Dependencies
 - Dependent relationships among components
 - Tool to model several dynamic reliability aspects of a generic system in the DRBD domain: inter and sequence dependencies, interferences, redundancy, load sharing, probabilistic correlated and common cause failures, reparability, ...
 - Represent a degree of freedom in the modeling of dynamic reliability behaviours with customizable policies and schemas



- A component is characterized by a variable *state*
- The state change is defined *event*
- The dependency parameter $\beta \geq 0$ characterizes the component under effect of a *dependency*





DRBD Solution Techniques and Algorithms

•Analytic

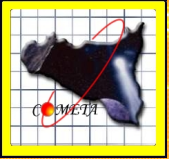
- 1)Static RBD structures equations
- 2)Markov Chain
 - ◆ Generally limited in reliability cdf modeling
- 3)Petri Nets
 - ◆ CPN with Aging Token, SPN, GSPN and NMSPN
- 4)Other techniques
 - ◆ Bayesian networks, ...

•Simulation

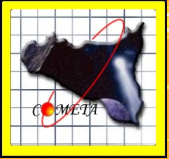
- 1)Monte Carlo
- 2)Other techniques
 - ◆ Discrete events, ...

•Combined Algorithm:

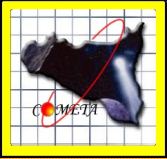
1. Identify static and dynamic independent parts
2. Static: use solution 1)
3. Dynamic: use one solution among 2), 3), 4), 5), 6) as convenience.
4. Integrate results (structures equations)



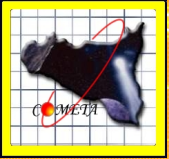
- **Generalization of the concept of dependency**
 - Composition features and management
 - Redundancy, Load Sharing, Interference, Probabilistic and Common Cause Failures, ...
- **Flexibility**
- **Extendibility**
- **More powerful than DFT and other derived or proprietary formalisms**
 - MDCS validates the DRBD methodology
- **DRBD Methodology studied on the Grid**



- ***Formal specification:*** a formal specification of the meaning of the DRBD will be developed.
- ***Petri Nets:*** developing of a DRBD solution technique based on *Non-Markovian Stochastic Petri Nets*.
- ***Simulation:*** developing of a DRBD simulation technique, to be used on the PI2S2 Grid infrastructure.
- ***Modeling and analysis tool:*** implementation of a tool to automatic analyze DRBD.
- ***Phased mission systems:*** definition of elements and rules to model this kind of systems.
- ***Reliability growth and fault coverage models:*** software reliability, non-perfect (coverage) maintenance.
- ***Case studies:*** Networking, Software, High Reliable Systems (Aircraft, Space, Underwater, Nuclear Plant, Chemistry...)

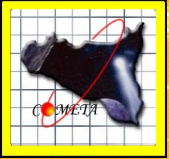


Software Performance Engineering: the PCM methodology

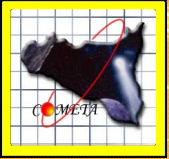


Software Performance Engineering

- **Early evaluation in design phase: bugs and performance issues**
- **Different approaches:**
 - Simulative
 - Analytic
 - Hybrid

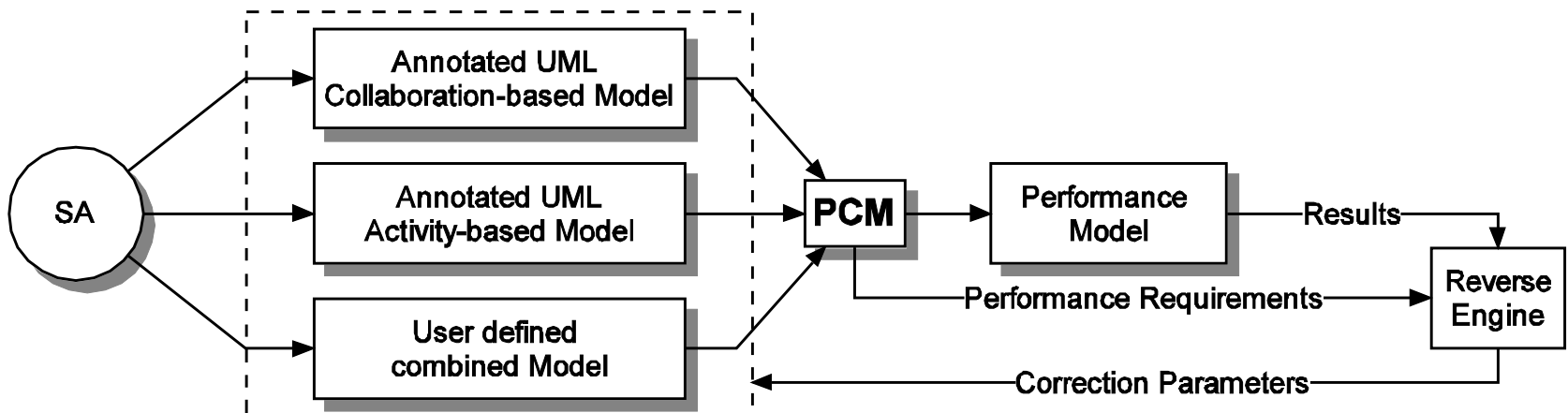


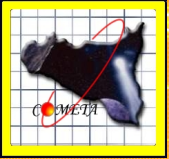
- In *Performance by Unified Model Analysis (PUMA)* D.Petriu et al, is proposed a tool architecture called *PUMA* which provides a unified interface between different kinds of software design tools (first and foremost, UML tools) and different kinds of performance models (Markov models, stochastic Petri nets ...).
- In *Simulation-Based Performance Modeling of UML Software Architectures* M. Marzolla, is proposed a description of system software through annotated *UML* diagram and a performance model as a *process-oriented* simulation model.
- In *A Software Performance Engineering Tool based on the UML-SPT* J.Merseguer et al, is proposed a tool that allows designing *UML* diagrams annotated according to the *UML-SPT*, and automatically generating a performance model in terms of *Generalized Stochastic Petri net (GSP)*.



From UML to NMPN: the PCM methodology

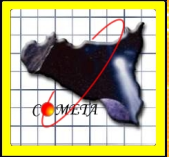
- A methodology to derive automatically performance measures of a system software.
- The solution is to split the transition from design domain (*UML*) to performance domain (*NMPN*) in two phases, interposing between them an intermediate model called: *Performance Context Model* (PCM).
- The *PCM* is an intermediate model between the *UML model* and the *Performance model*, and allows to adopt any approach (Simulative, Analytic, Hybrid) in order to estimate the performances of a Software Architecture (SA).





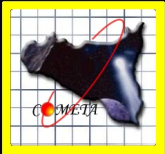
2 fundamental steps

- The overall process could be synthesized into two steps:
 - UML Source Model → PCM
 - PCM → Performance Model
- During step 1 all the information about the software project, including the performance annotations (according to the *UML-SPT*), will be formally specified in the UML model.
- A solvable performance model, (we characterize as the Petri nets domain) will be obtained through step 2.



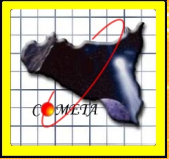
Different views

- In the approach a formal methodology to represent a *Software Architecture (SA)* by a *UML* model is also defined: the *UML3VIEW*.
- It is a representation based on three level:
 - *Global View*: describes the general behaviour of a SA through *UML Use Case Diagram (UCD)*.
 - *Specification View*: characterizes the SA in low level, describing the workload associates to the SA through UCD and the resource dislocation through UML Deployment Diagram (DD).
 - *Detailed View*: defines the dynamic behaviour of a SA, using a *UML Activity Diagram (AD)*, a *Sequence Diagram (SD)* and a *Collaboration Diagram (CoD)*.



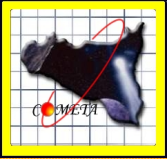
The following tools have been developed:

- *ArgoPerformance* is a plugin for *ArgoUML* tool. Its intent is to permit the adding of performance annotations in UML diagrams to forecast performance problems from the early stages of Software Development Process.
- *WebSPN* is a new modeling tool for the analysis of non-Markovian stochastic Petri nets. A Grid version has also been developed.

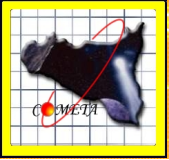


Future work

- **Studying if performance analysis can be performed directly from the PCM model**
- **Implementation of the PCM model as a event-driven simulator**
- **Simulation on the PI2S2 Grid infrastructure**



- **UniMe among its activities is working on**
 - security on Grid
 - enhancement of the Grid data storage is currently undergoing
 - conformity to the actual standards
 - dependability evaluation of dynamic systems
 - a new approach is under development: DRBD
 - study is conducted using the PI2S2 Grid infrastructure
 - software performance engineering
 - a new approach is under development: the PCM methodology
 - tools have been developed
 - argoperformance
 - WebSPN (a grid version is also available)
 - PCM will be ported on the PI2S2 Grid infrastructure



- **Prof. Antonio Puliafito, email: apuliafito@unime.it**
- **Prof. Marco Scarpa, email: mscarpa@unime.it**
- **Dr. Carmelo Ragusa, email: cragusa@unime.it**