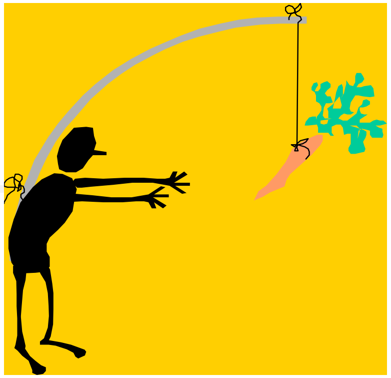
A Model-driven Approach to Predictive Non Functional Analysis of Component-based Systems

Vincenzo Grassi and Raffaela Mirandola

Università di Roma “Tor Vergata”, Italy

Goal



Architecting systems with components with predictable quality of services

Performance

performance, reliability, availability...

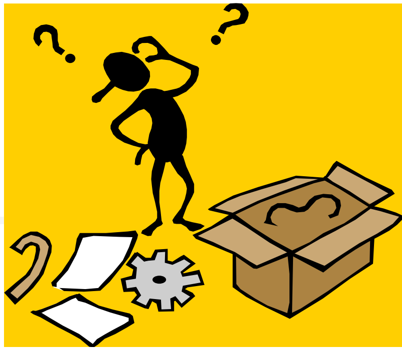
Software component

What is a software component?

"A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties"

C. Szyperski

Starting point

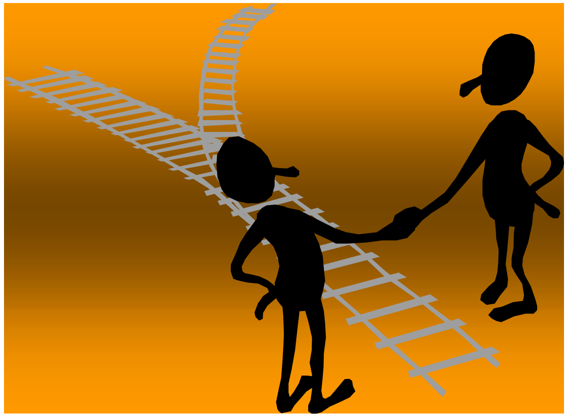


Heterogeneity

- ✓ Hardware platforms
- ✓ Operating systems
- ✓ Network protocols
- ✓ Programming languages

Looking for consensus

Based on MODELS



Model Driven Approach

model definition:

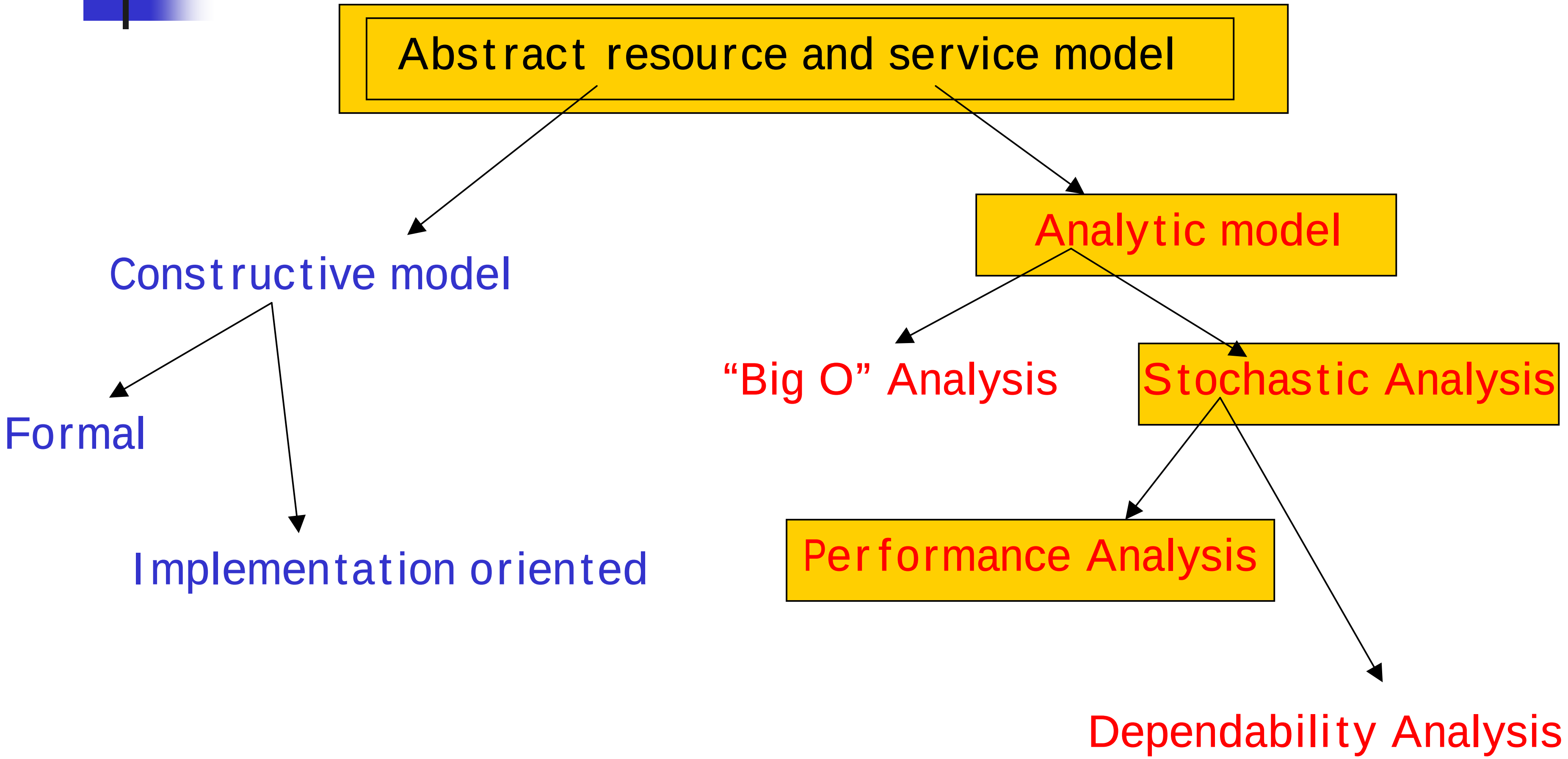
sequence of *refinement steps*

each step specializes and enriches a more “abstract” model defined at the previous step

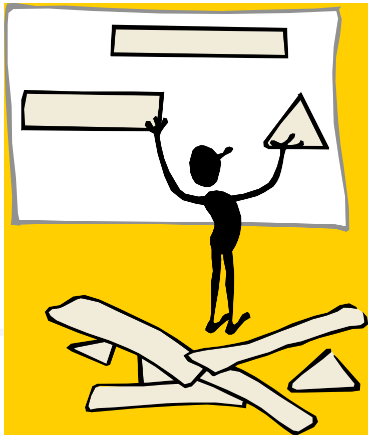
Isolation and understanding of basic concepts that must be modeled and their interdependencies

at each step different refinements can be devised (definitions of different specialized views of the same system)

Model Driven approach



Definitions



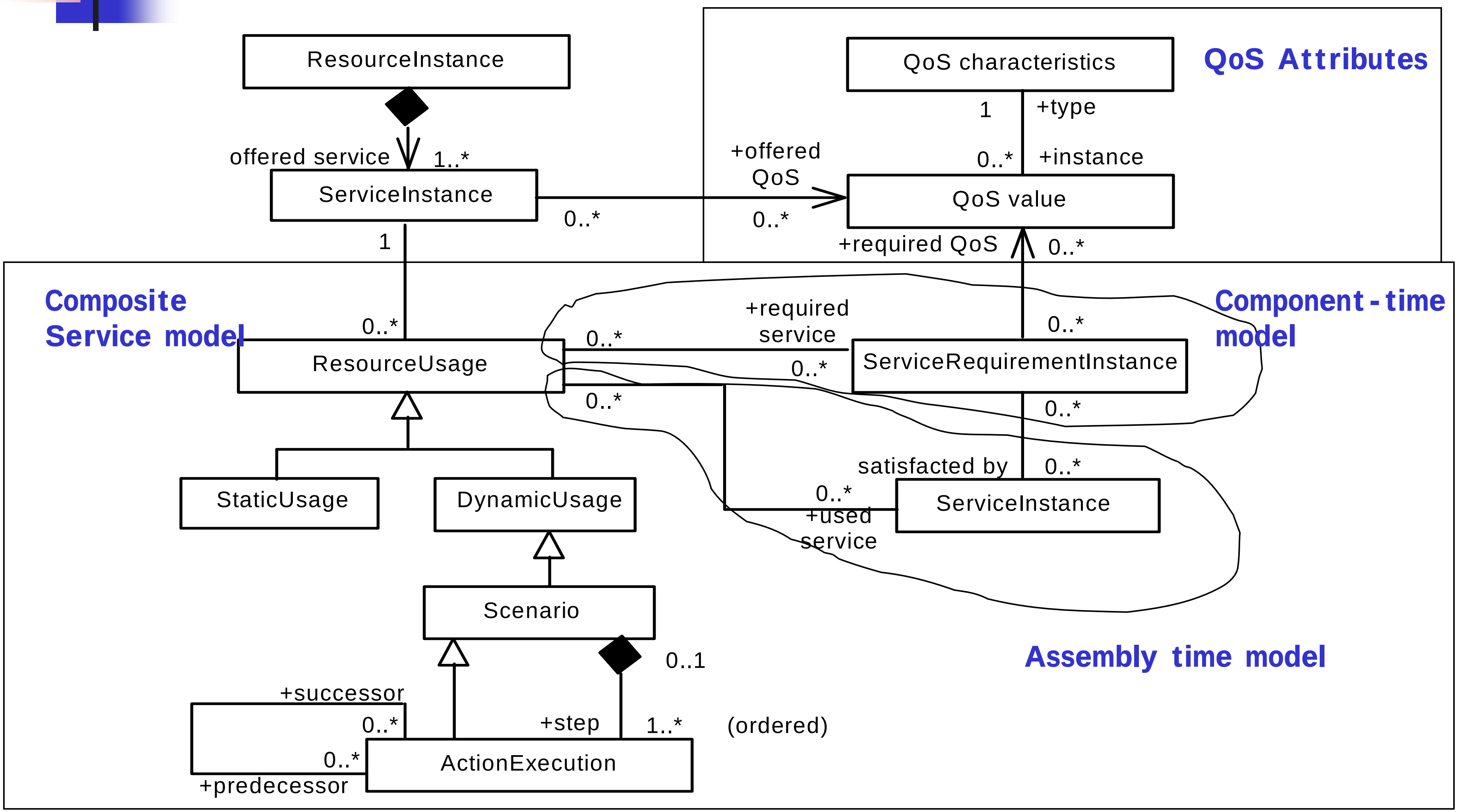
By ***resource*** we mean any run-time entity offering some service
- software components
- physical resources like processors, communication links or other devices

a ***service*** can correspond to some "high level" complex task, or some "low level" task such as the processing service offered by a processor.

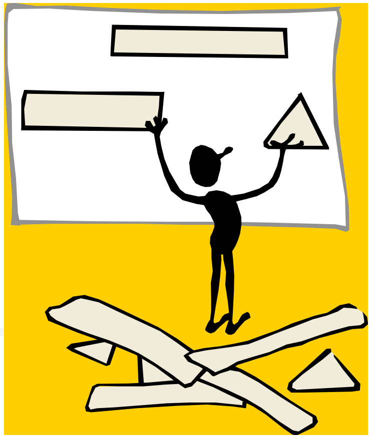
simple services that do not require any external service to carry out their task,

composite services, that instead do require them

Root model (GRM-like)



...definitions



component time service model,

where the required services are specified through a set of constraints that characterize them

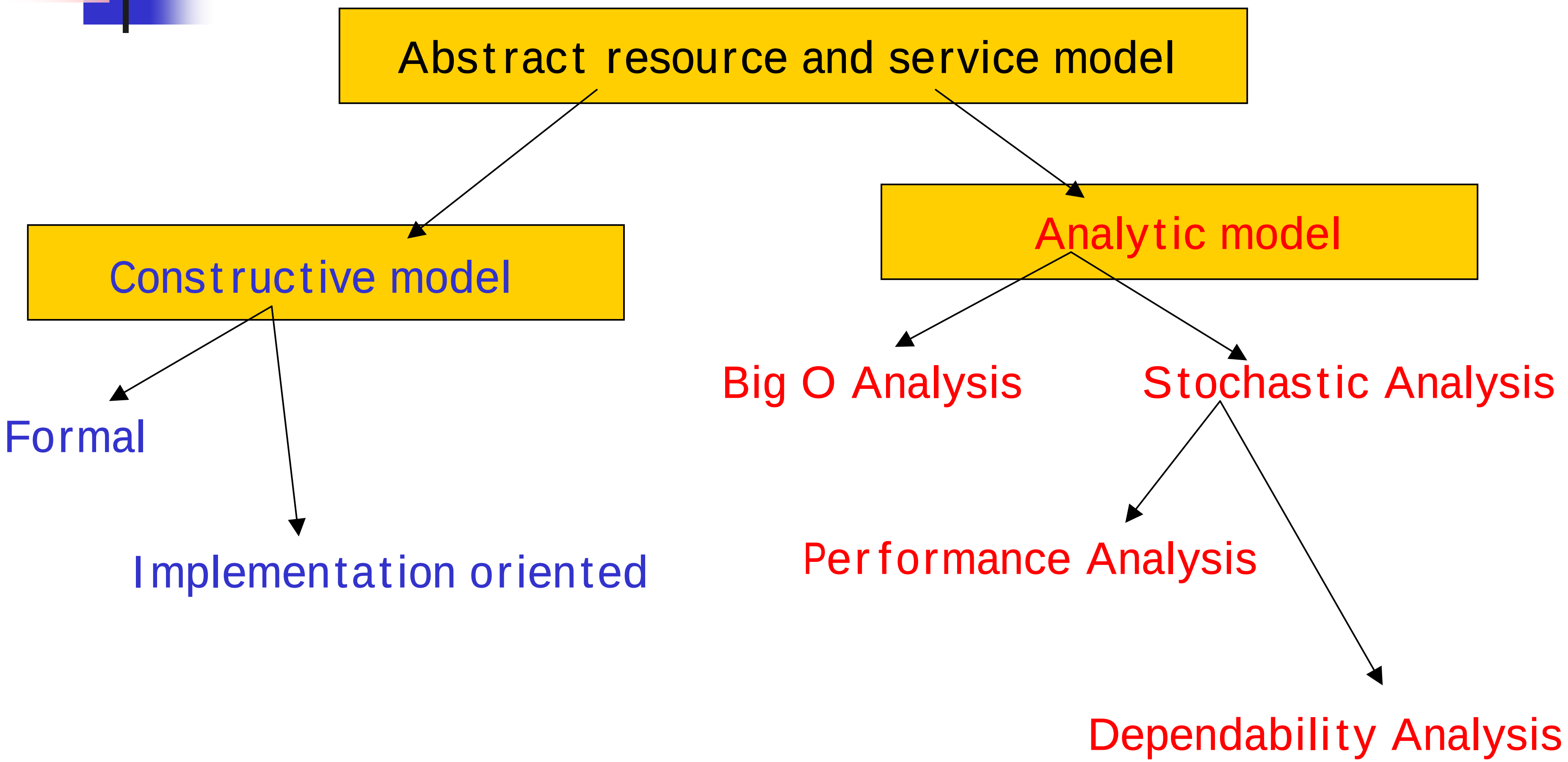
assembly time service model

where the service is actually linked to service instances that satisfy those constraints

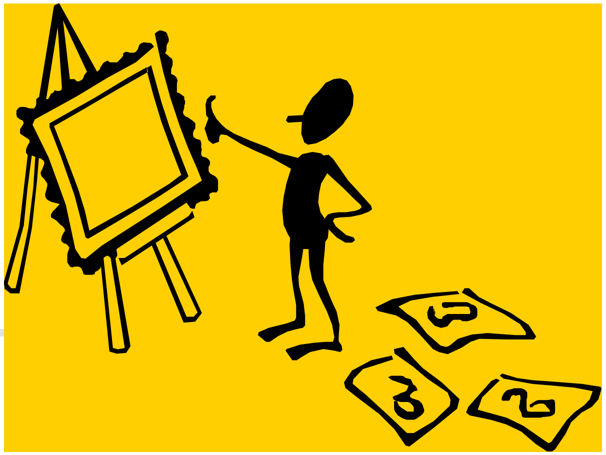
dynamic service usage model

specify some pattern of use of the required services: specification of action (a specific instance of an invocation of some required service) executions

MDD


Abstract resource and service model
Constructive model
Analytic model
Formal
Implementation oriented
Big O Analysis
Stochastic Analysis
Performance Analysis
Dependability Analysis

Constructive …



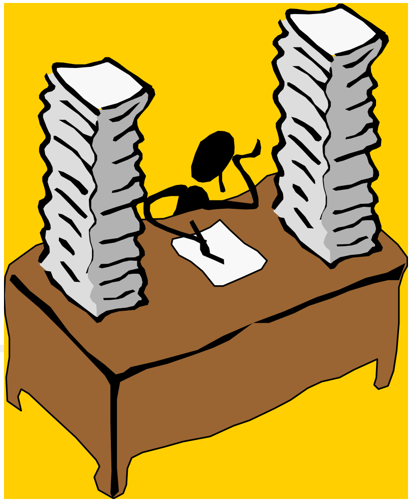
…refinement:

- *service*: specification of a “constructive” interface (e.g. the service signature: name and data type of the formal parameters)

- *scenario*: specification of pattern of “activities”, expressed using C-like control constructs (conditional statements, loops)

- *action execution*: specification of values of the actual parameters for external required services invocation

… vs. analytic refinement

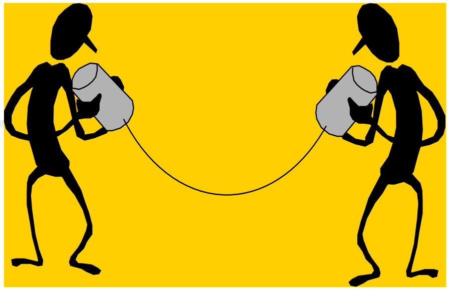


➢*service*: specification of an “analytic” interface (e.g. name and set of values of the formal parameters)

➢*scenario*: specification of a pattern of “activities” expressed using some stochastic model (e.g. probabilistic execution graph, stochastic Petri net)

➢*action execution*: specification of random variables modeling the values of the actual parameters of a service invocation (these random variables must take values in the set of values for the corresponding formal parameter)

Constructive vs. analytic refinement: "abstraction mapping"



service:

"constructive" ➔ "analytic" formal parameter

e.g. partitioning the original domain into a (possibly finite) set of disjoint sub-domains, and then collapsing all the elements in each sub-domain into a single representative element

scenario:

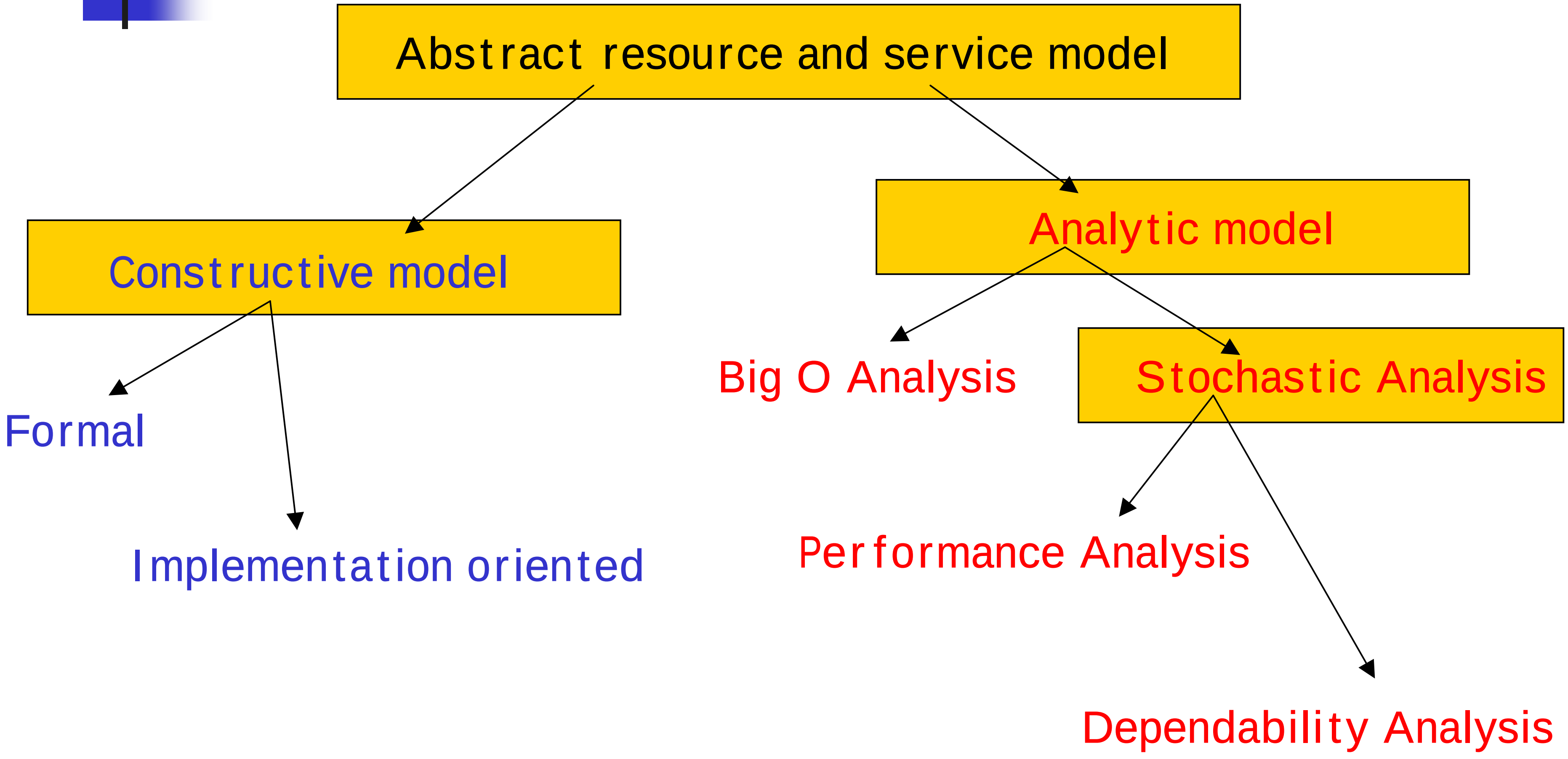
e.g. conditional statements become probabilistic selections of alternative paths

action execution:

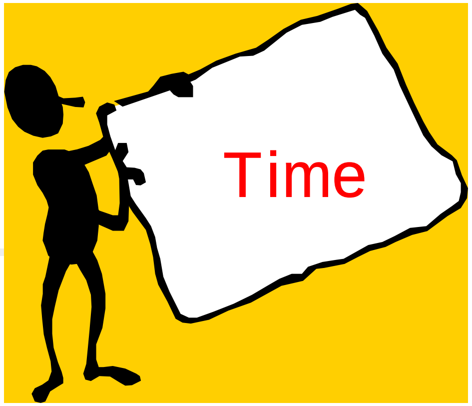
constructive ➔ analytic actual parameters

e.g. the probability distribution of the adopted random variables is representative of the actual distribution of values in the constructive parameters.

MDD


Abstract resource and service model
Constructive model
Analytic model
Formal
Implementation oriented
Big O Analysis
Stochastic Analysis
Performance Analysis
Dependability Analysis

Stochastic model refinement



timeliness aspects of a system

"provided QoS" attributes: $T_{exec}(i)$

time taken to carry out a single request for an offered service *Si*

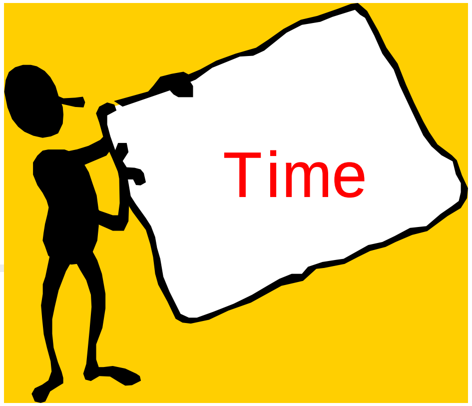
In a stochastic setting, $T_{exec}(i)$ is specified by a random variable

parametric with respect to the service input parameters

whether the service is a *simple* or *composite* service;

whether the service is a *no contention* or *contention-based* service.

Stochastic model refinement(2)



$$T_{exec}(i) = T_{int}(i) + T_{cont}(i) + T_{ext}(i)$$

Component time

$T_{int}(i)$: time spent in internal actions

$T_{cont}(i)$: time spent waiting before actually accessing the service

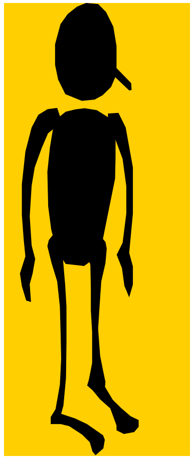
Assembly time

$T_{ext}(i)$: time to carry out externally required services

with: $T_{ext}(i) = \bigoplus_{Sj \in Required(Si)} T_{exec}(j)$

Stochastic model refinement(3)

- *contention unaware*:



$T_{cont}(i) = 0$ for all services, that corresponds to assuming that all services are no contention services

model for the calculation of $T_{exec}(i)$ uses only information associated to the dynamic resource usage of each assembled service *Si*, neglecting any contention or access control issue (e.g., "connection" of the execution graphs of the assembled services) and graph analysis techniques to calculate the overall completion time.

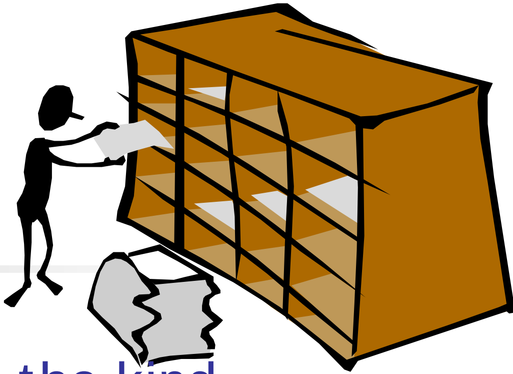
- *contention aware*:

$T_{cont}(i) \geq 0$

Example: sort and search service

a resource that offers a search service for an item in a list; to carry out this service, it requires a sort service (to possibly sort the list before performing the search) and a processing service (for its internal operations). In turn, the sort service requires a processing service.

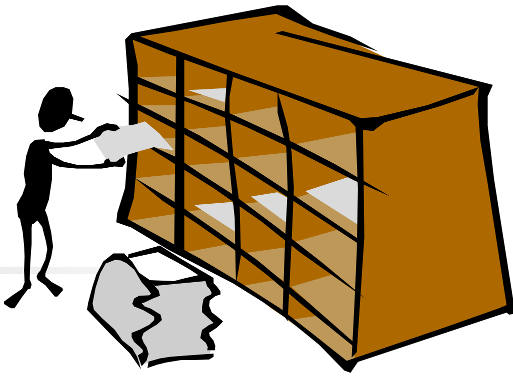
Basic GRM-based model



identify the resources involved in the application and the kind of offered and required services with their basic characterization

Resource	Offered services	Service type	Required services
Search_res	search(list, item)	composite	process, sort
Sort_res	sort(list)	composite	process
CPU_res	process(op_list)	simple	none

Constructive refinement



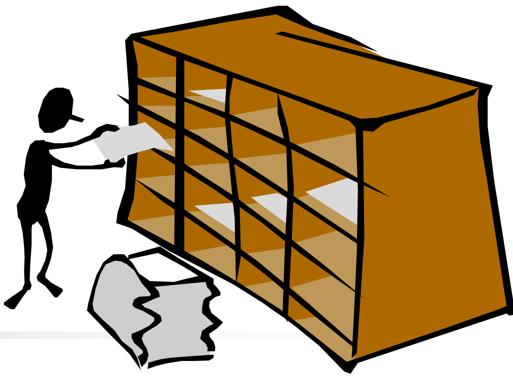
Composite sort and search service characterization

```
Sort_res.sort(l:list of T) =
        {call(process(sort_algorithm(l)))};
Search_res.search (l:list of T, i:T) =
        {if (not_ordered(l)) call(sort(l));
           call(process(search_algorithm(l)));
           }
```

processing service characterization

```
CPU_res.process(oplist:list of MachineOperation) =
      {do(oplist)}
```

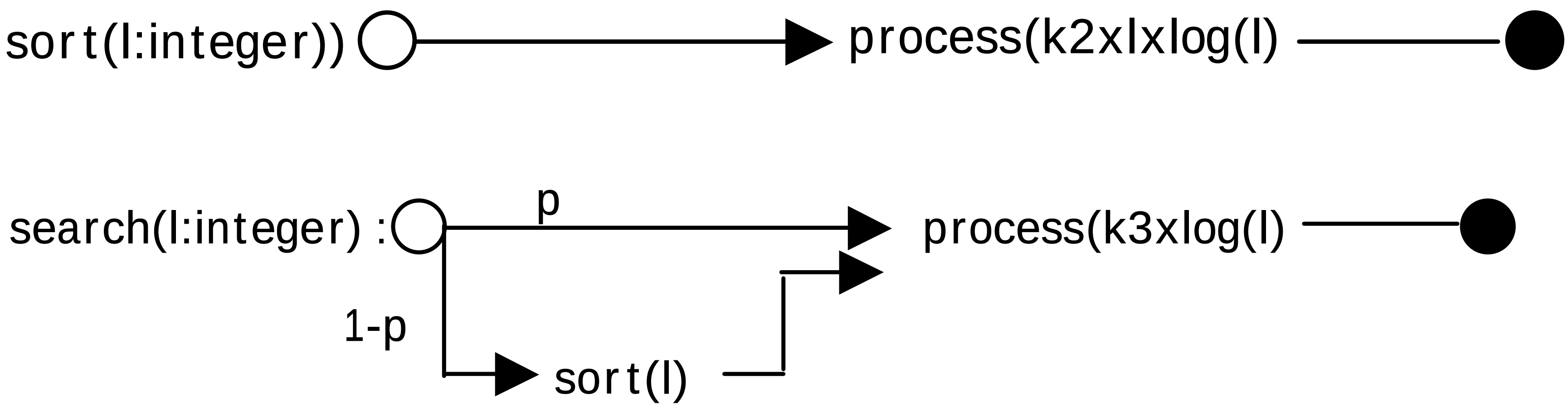
Analytic refinement: stochastic approach



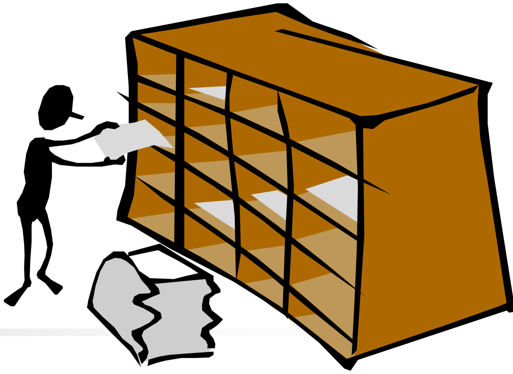
Characterization of search and sort services:

the list formal parameter could be defined as l:integer, with domain given by the set of non negative integers, each representing the size of some list

pattern of activities of the sort and search composite services:



Analytic refinement: stochastic approach



actual parameters

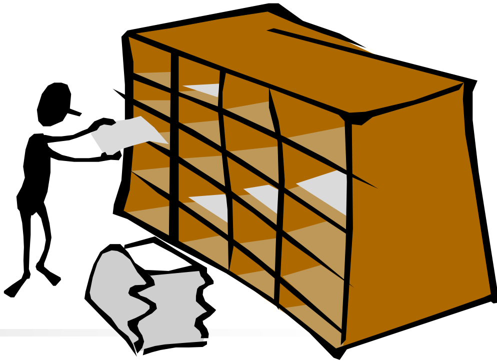
random variables parametric with respect to each service formal parameters (l)

For a quicksort algorithm:

the actual parameter for the process request
can be modeled as an integer valued random variable
in the range $[k1 \times l \times \log(l), k1 \times l^2]$,

Characterization of the process service:
an entity executing a single kind of “average” operation
with a formal parameter defined as oplist:integer that specifies the number
of such operations

Contention unaware analysis



For this kind of analysis we assume the $T_{cont} = 0$

$$T_{exec}(\text{process}(\text{oplist})) = T_{int}(\text{process}(\text{oplist})) = \text{oplist} / \text{cpu\_speed}$$

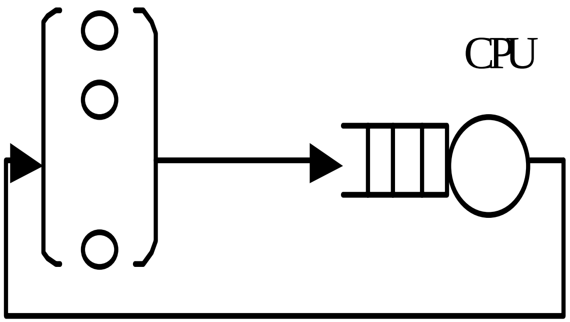
$$T_{exec}(\text{sort}(l)) = T_{ext}(\text{sort}(l)) = T_{exec}(\text{process}(k2 \times l \times \log(l)))$$

$$T_{exec}(\text{search}(l)) = T_{ext}(\text{search}(l))$$
$$= T_{exec}(\text{process}(k3 \times \log(l))) + (1-p)\, T_{exec}(\text{sort}(l))$$

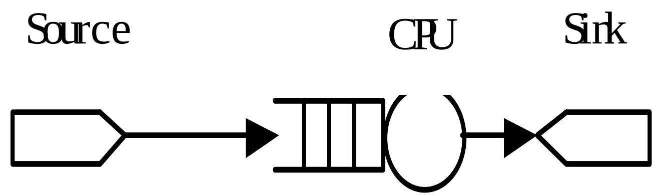
Finally:

$$T_{exec}(\text{search}(l)) = \boxed{k3 \times \log(l) / \text{cpu\_speed}} + \boxed{(1-p) k2 \times l \times \log(l) / \text{cpu\_speed}}$$

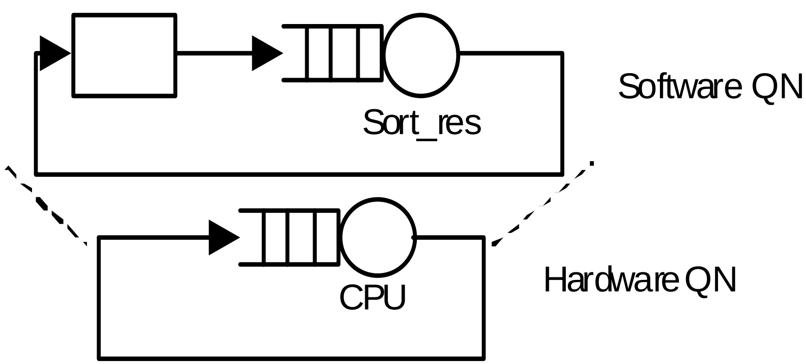
Contention aware analysis



CPU is a contention resource

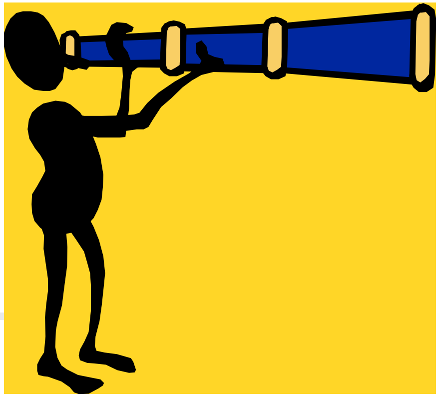


CPU and Sort are contention resources



$T_{exec}(\text{process(oplist)}) = T_{int}(\text{process(oplist)}) + T_{cont}(\text{process(oplist)})$

Conclusions and future work



Definition of a path that leads to the construction of a stochastic model for the compositional performance analysis of component-based systems

actual “implementation” of this path

definition of a suitable language to express the needed information

with a precisely defined syntax and semantics that support the development of automatic tools for QoS predictive analysis of component-based systems