Modelling for security and dependability

Marco Domenico Aime

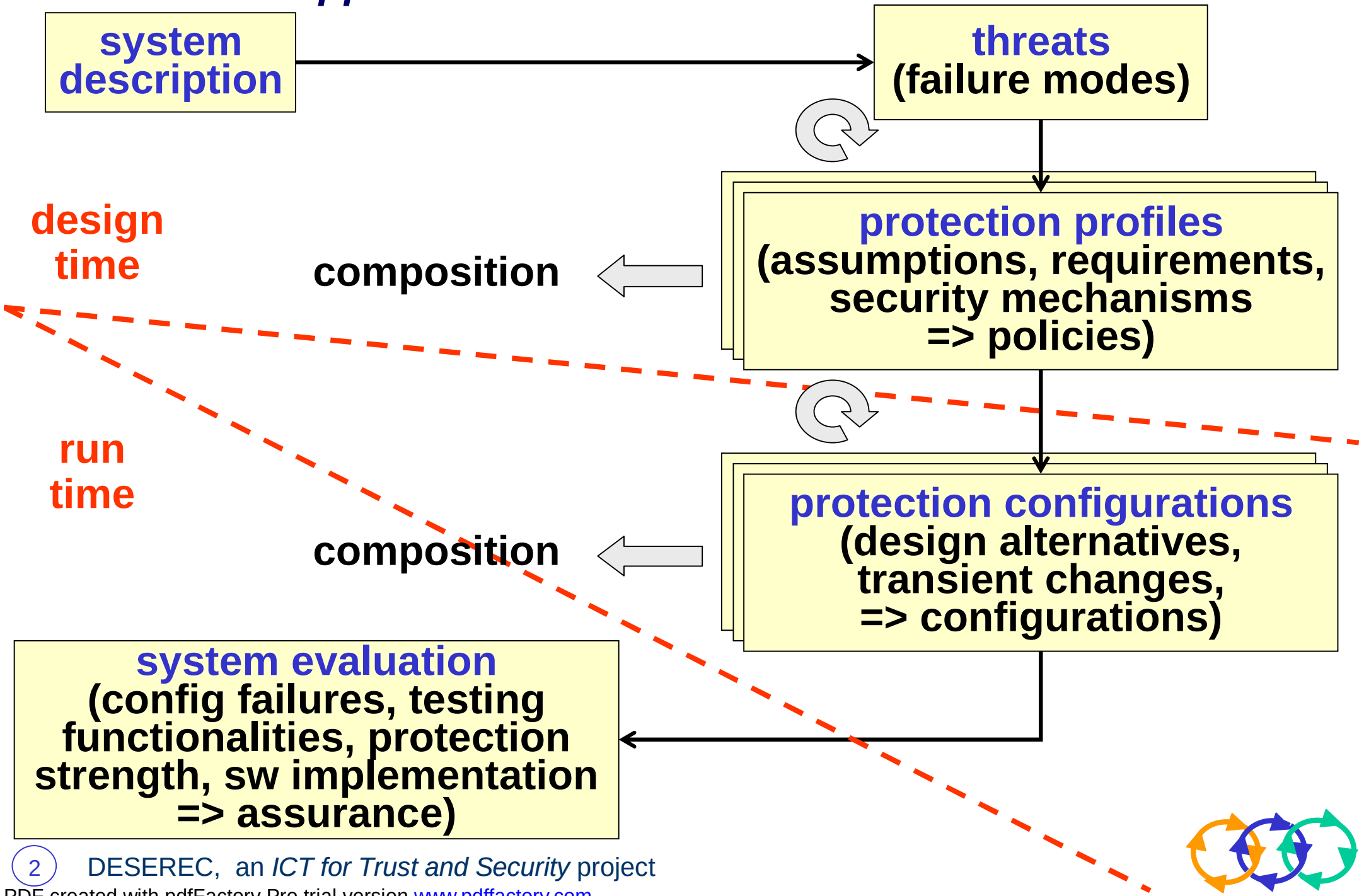
POLITECNICO DI TORINO

26th September 2006
Wroclaw

DESEREC

Dependability and Security by Enhanced Reconfigurability

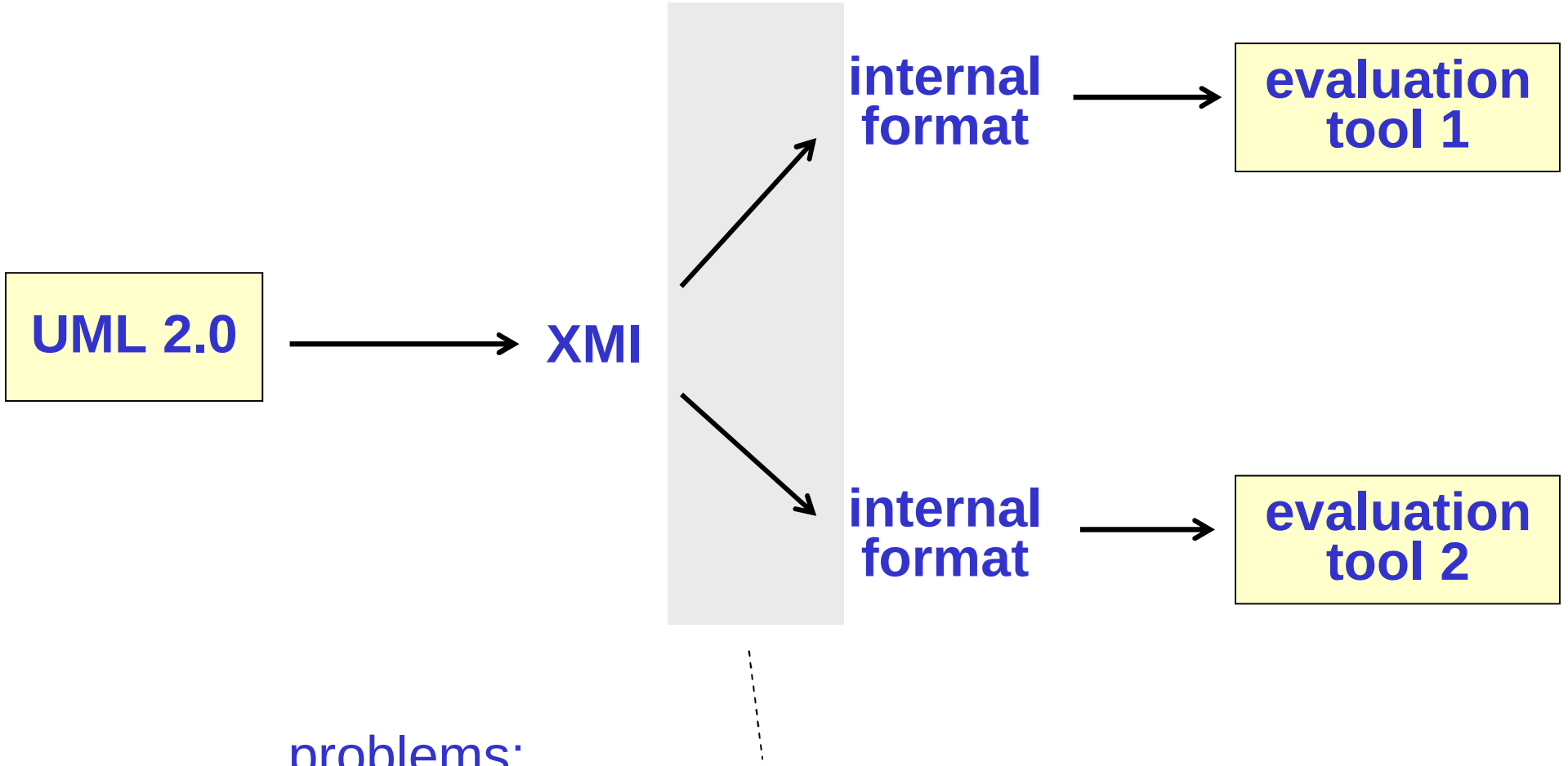
Model-based approach



Model-based system analysis

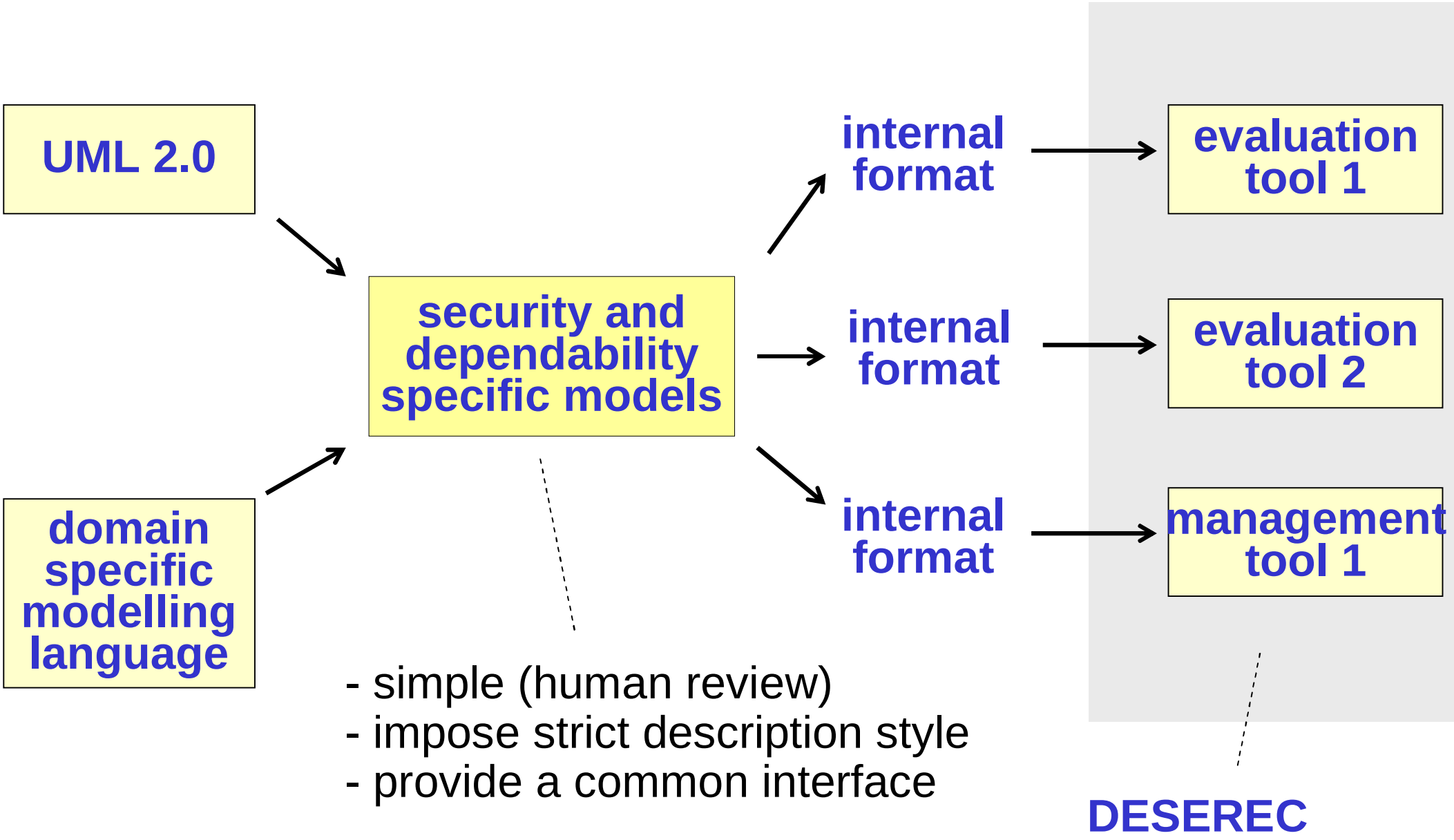
- threat analysis:
  - vulnerability analysis
  - risk assessment
- detect conflicts in models (and propose resolution):
  - conflicts within requirements
  - conflicts within security & dependability rules
  - conflicts between requirements and rules
- compare design alternatives:
  - policy refinement
  - enforceability of rules in the system
  - system survivability (accidental faults and attacks)
- plan system management:
  - component configuration generation
  - incident detection planning
  - incident reaction planning

A typical approach



problems:
- “this isn’t the UML I like…”
- consistency and completeness of translation?

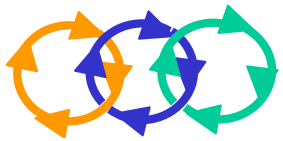
A practical approach



Requirements for system modelling 1/2

system description:

- hardware: hw resources and their aggregation
- software: software resources and their aggregation
- services: their composition, their interaction (workflow)
- network: network topology, physical and logical, addresses
- information: storage and flow
- security, dependability, QoS features
- environment: locations and support systems (power supply)



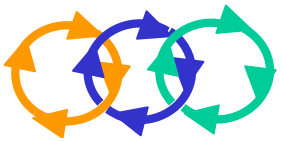
Requirements for system modelling 2/2

policy description:

- requirements and constraints (protection, QoS)
- security and dependability mechanisms, and their configuration (rules)

relationships:

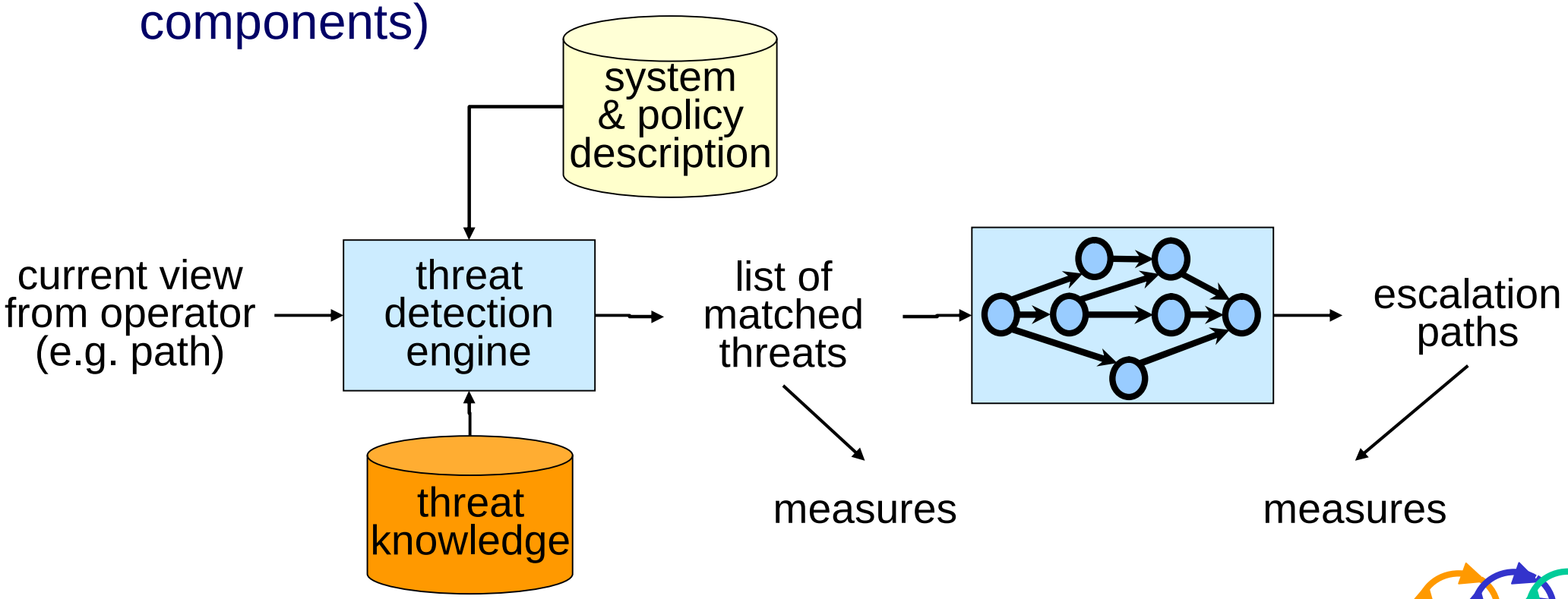
- physical allocation of services
- security and dependability rules mapping
- relationship between multi-layered descriptions (refinement)
- information processing



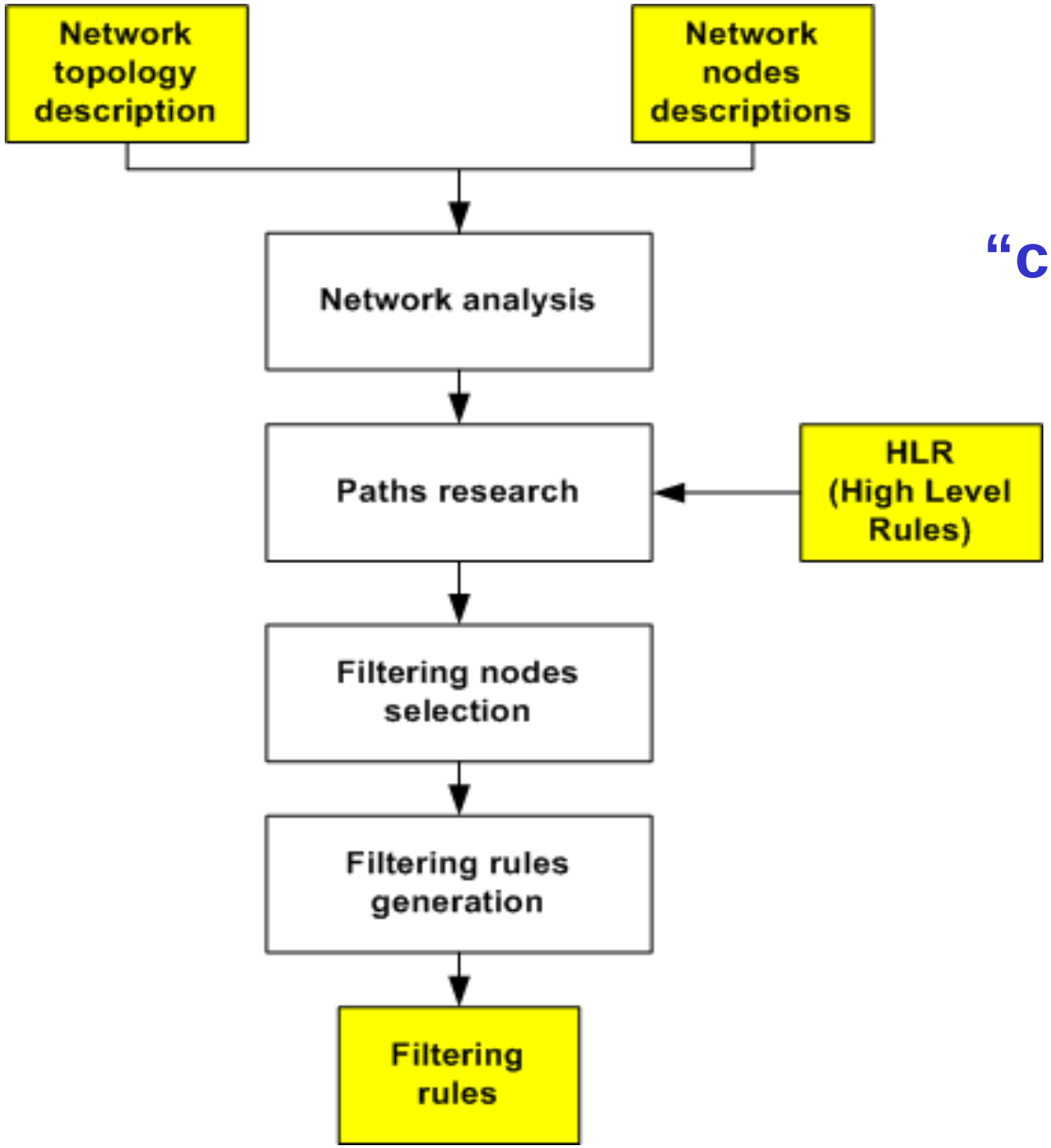
Exmaple: threat analysis

Model-based threat analysis:

1. find threats given a system description
2. measure threats (based on affected system components)
3. find threats escalation paths
4. measure threats escalation paths (based on affected system components)



Example: connectivity policies refinement

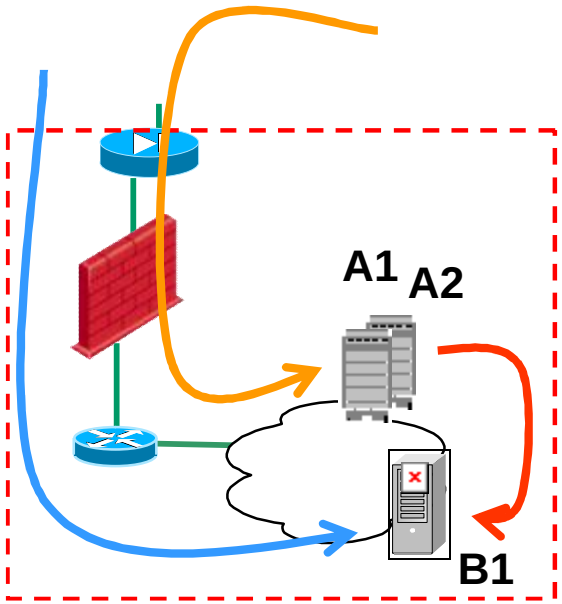


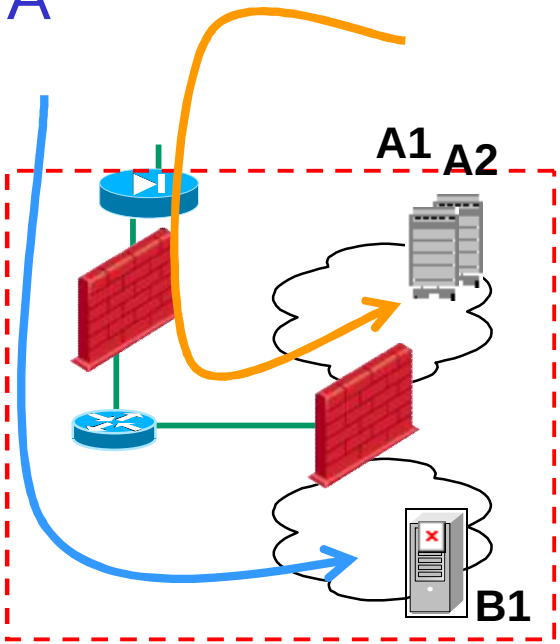
“connect DAUIN with Internet”

Example: rules enforceability

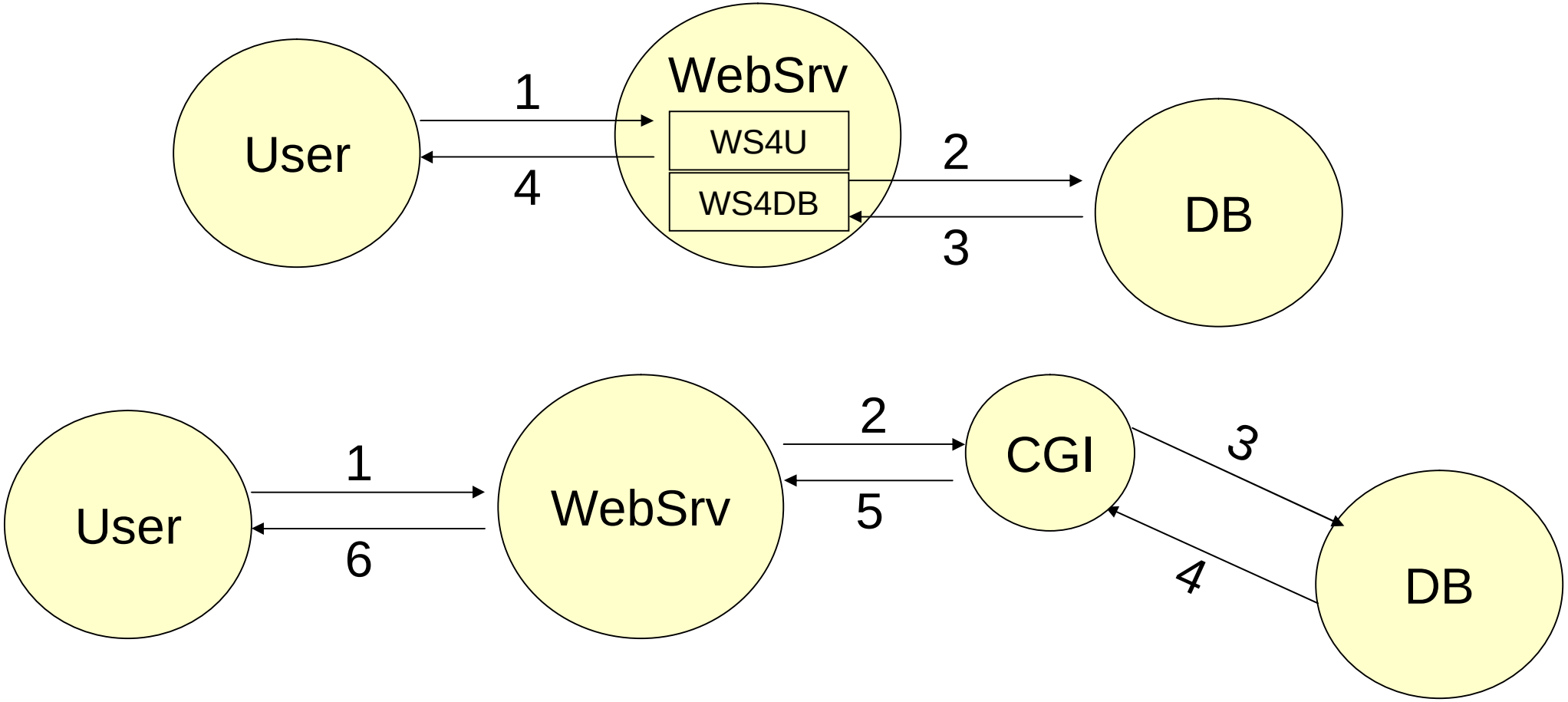
- disconnect two nodes:
  - impossible if there is no filtering device between them
  - graph-based analysis
  - integrate with high-level connectivity policies tools

disconnect B* from A*



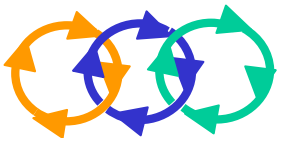


Example: description at different levels of details



Examples: service oriented dependability evaluation

- static analysis:
  - find a design bug (e.g. password authentication over insecure protocol)
  - find where we could be attacked (e.g. data flow, roles)
  - find functional dependencies (e.g. attack/faults effect on availability)
  - find attack paths at application level (e.g. attack to a DB through the frontend)
- dynamic analysis:
  - find policies violating service constraints (e.g. authorisation)
  - find constraint violations starting from a property violation
  - find attacks against the workflow (e.g. triggering exceptions)
- compare design alternatives:
  - compare different service compositions (e.g. payment solutions)



Find a design bug

no problem if:

- username/password is flagged confidential
  - constraints = confidential (from the semantic information)
- the protocol between User and Service preserves confidentiality and integrity
  - security properties = preserves confidentiality, integrity (from the resource view and allocation information, e.g. https)
- the protocol between Service and AuthService preserves confidentiality

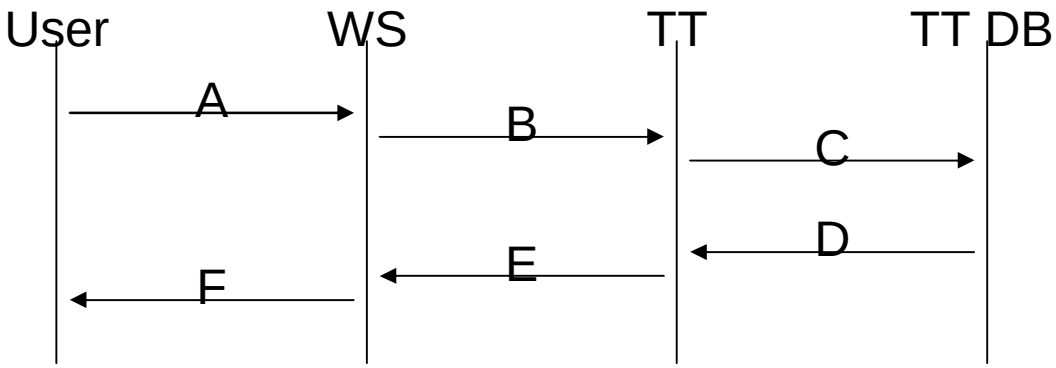
but we have a possible confidentiality loss changing to:

- the protocol between Service and AuthService does NOT preserve confidentiality (e.g. unprotected SQL access)

Find where we could be attacked

potential loss of integrity for the TimeTable response:

- on the TimeTable DB
- in the exchange D
- on the TimeTable server
- in the exchange E
- on the WebServer
- in the exchange F

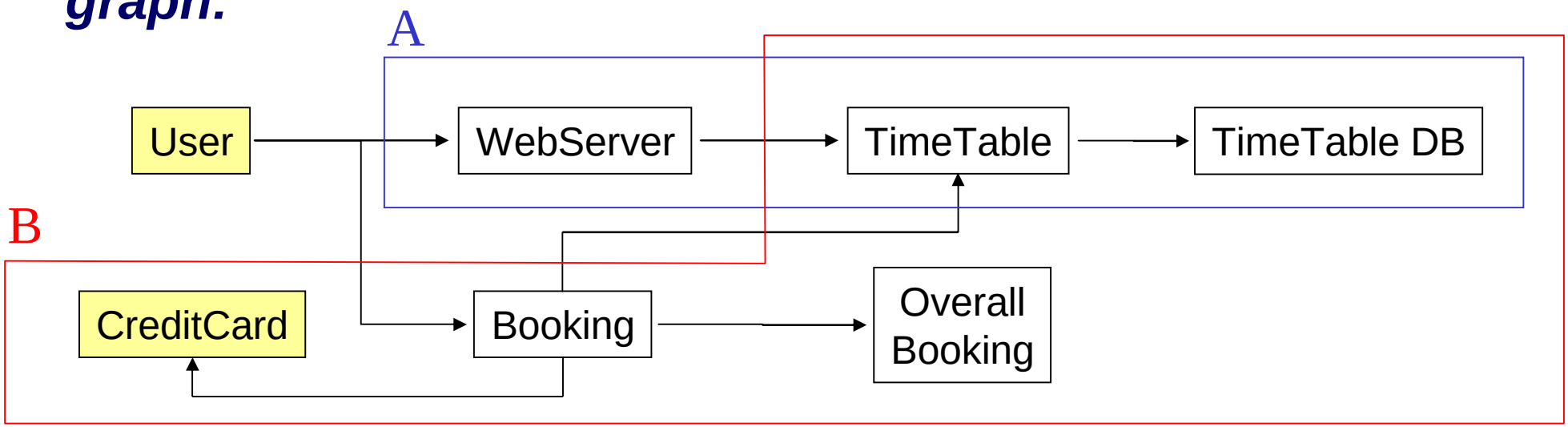


using information from resource layer and allocation:

- if the protocol between User and WebServer preserves integrity, then exchange F would not be prompted by the analysis
- if TimeTable and TimeTable DB actually run on the same machine, attacks against exchange D may not be feasible and attacks on TimeTable DB and TimeTable collude

Example: find functional dependencies

from interaction between the services derive an oriented graph:



- WebServer, TimeTable or TimeTable DB unavailability result in a failure on the deployment of the service
- WebServer is directly accessible from the outside

may apply also to multiple services

- TimeTable is a single point of failure for either service A and B
- this holds for everything TimeTable depends on (TimeTable DB)

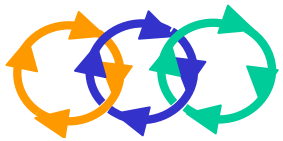
Find attack paths at application level

find flows that an attacker can use to reach his target

- then analyse them with the methods presented before
- may compose multiple service flows (i.e. taking advantage of a service to attack another one)
  - e.g. same credential used for different services

using information from resource layer and allocation

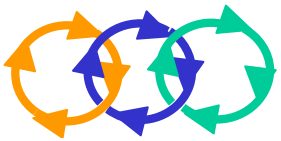
- data flow analysis (confidentiality and integrity)
  - select the network path (resource view) critical data flows through (workflow view)
  - verify a choreography channel tagged “secure” (workflow view) is allocated to “secure” links or protocols (resource view)
- application-level reachability (availability)
  - verify that firewalls rules (resource view) prevent other accesses than what prescribed in the workflow view



More on multi-level descriptions

consistency checks:

- correctness:
  - verify that everything is in the service description actually exists in the topology description
  - verify that security properties specified in the service description are satisfied in the topology description
- completeness:
  - verify that nothing in the service structure description has been forgotten (i.e. if something is not possible in service description, then it is not possible in the topology description)



Dynamic analysis

- what-f analysis
- simulate the workflow

find policies violating service constraints

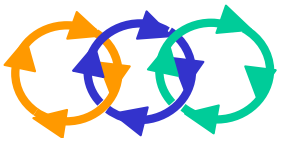
- e.g. changes in the authorisation policies

find constraint violations starting from property violation

- e.g. data confidentiality loss, if SSL long term credentials are compromised

find attacks/faults against the workflow

- suppose faults/attacks in the topology description and look for service workflow exceptions
- e.g. service gets stuck if sub-service not responding, no user notification



Compare design alternatives

analyze different implementation choices by:

1. building different serive choreographies, one for each implementation
2. analysing all the service choreographies following the same procedure
3. comparing the results

- example: authentication credentials:
  - username/password credentials are confidential data, whereas the data exchanged in a X.509 client based authentication is not
  - hence our analysis should prefer X.509 certificates in an environment where insecure protocols exist

Compare design alternatives

- example: credit card payment for booking:
  1. AuthService stores user data including CC number. Service requests to AuthService the CC number once the user is authenticated, then requests a confirmation to User, and finally contacts CreditCard service to perform the payment;
  2. Service requests to User his CC number once he is authenticated, then contacts CreditCard service to perform the payment;
  3. Service authenticates User, then redirects him to towards CreditCard service; User perform the payment, and CreditCard confirm the payment both to User and Service (e.g. Paypal)
- analysis = decreasing number of “places” for CC’s confidentiality loss:
  1. confidentiality can be lost on AuthService, on Service and in every exchange;
  2. confidentiality can be lost on Service and in the exchanges;
  3. no confidentiality loss should occur (from the Service point of view)