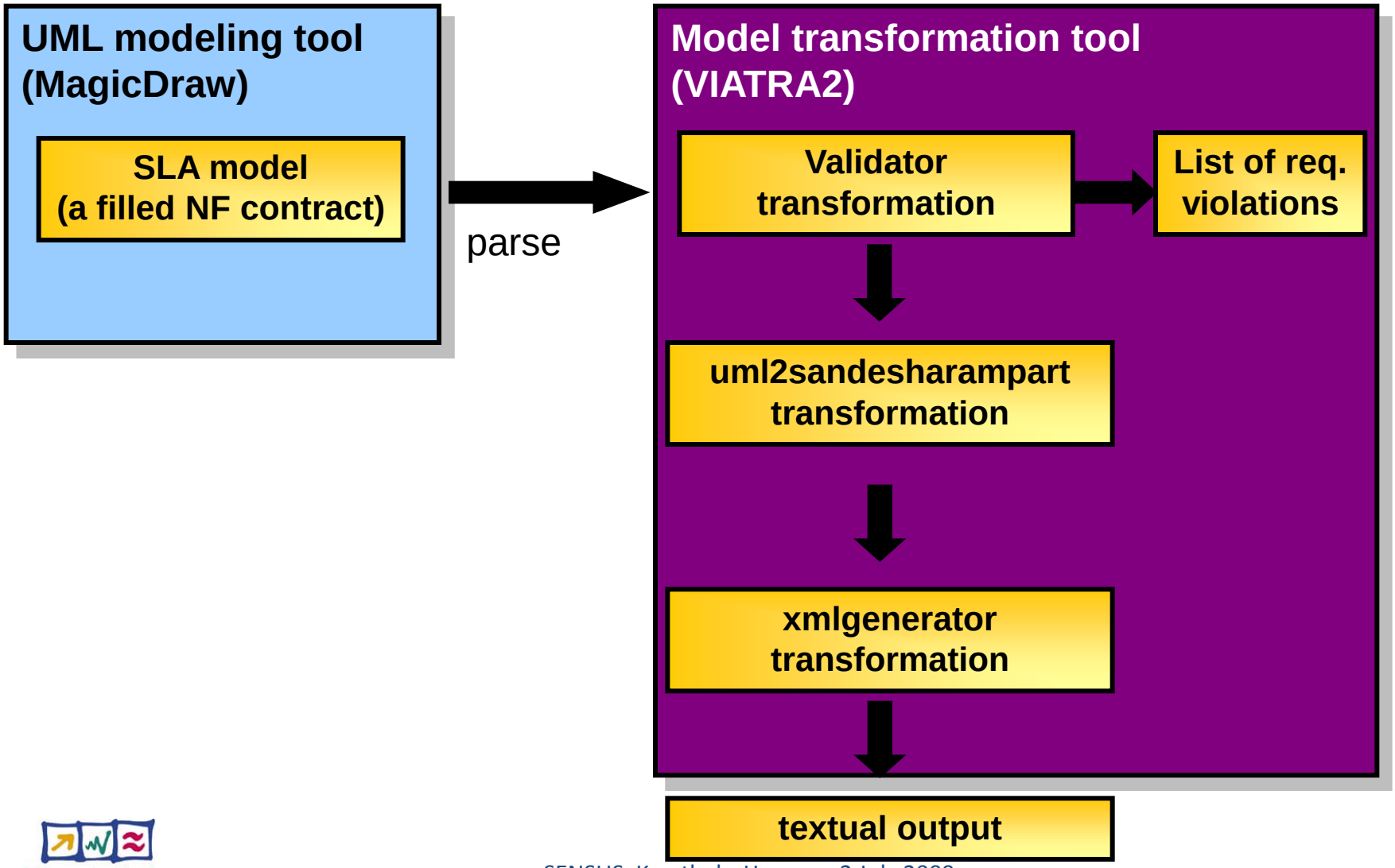
Model driven deployment lab

László Gönczy, Dániel Varró (BUTE)
gonczy@mit.bme.hu


UML modeling tool (MagicDraw)
SLA model (a filled NF contract)
parse
Model transformation tool (VIATRA2)
Validator transformation
List of req. violations
uml2sandesharampart transformation
xmlgenerator transformation
textual output

What will you do in the lab?

- Follow a deployment process on the Financial Case Study example
- Create an own SLA template and SLA model for the eUniversity
- Generate „pseudo”code from the model
- Extend the validator transformation

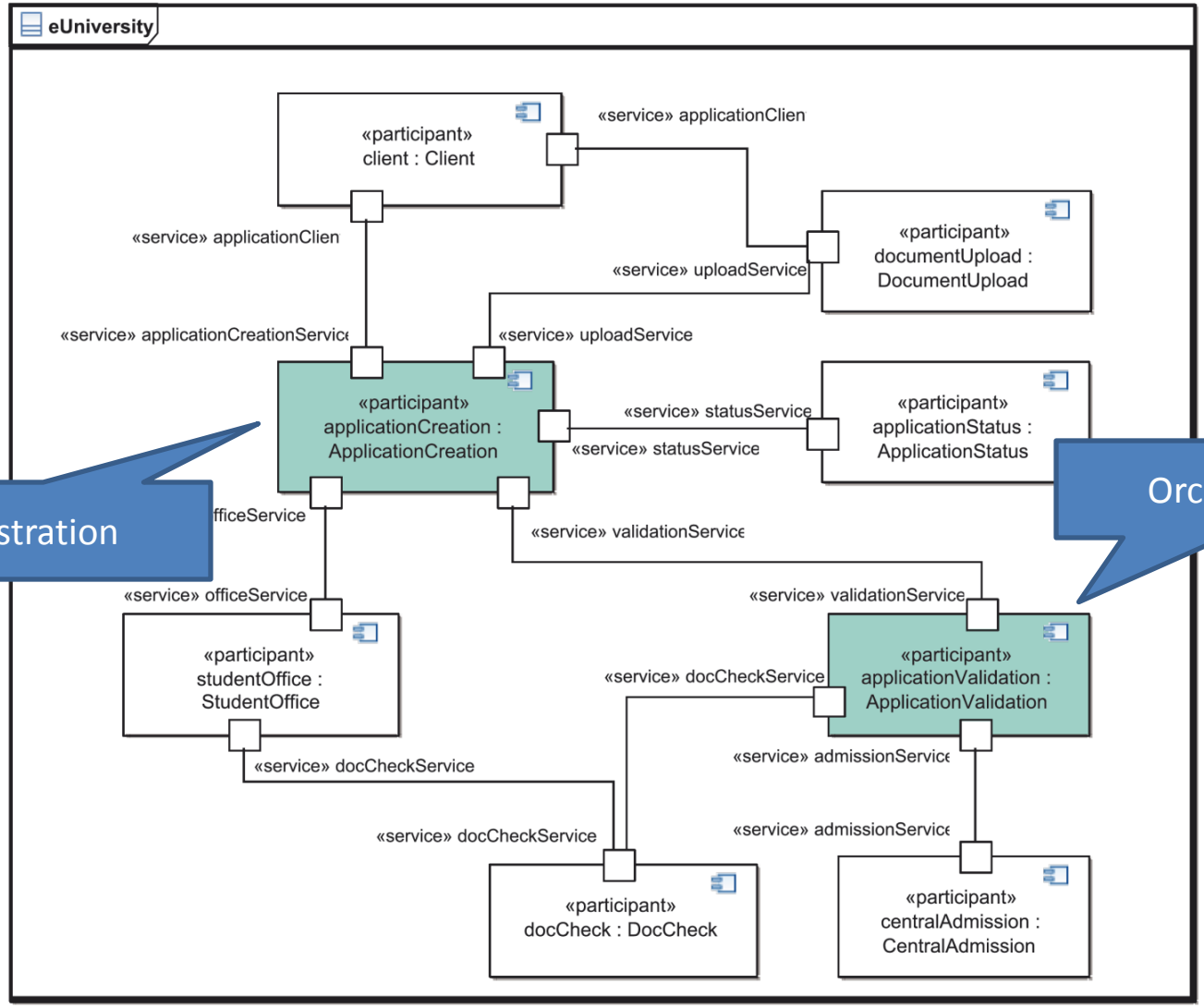
- Our non-functional modelling approach was validated using the Sensoria **eUniversity Case Study**, modelled in UML4SOA.

- Chosen scenario: **student registration**
  - A student wants to apply for a certain course of study at an online university
  - He provides all his data (grades, documents, bio, ...) via a web site to a web service (actually, an UML4SOA orchestration).
  - This orchestration checks the application with the help of other web services and orchestrations.
  - The result is presented to the student online as well.

- Two orchestrations + six atomic web services
  - Orchestrations: **ApplicationCreationService** and **ApplicationValidationService**
  - Web Services: **Client, UploadSystem, AppStatus, StudentOffice, DocCheck, CentralAdmission**

eUniversity: Modelling

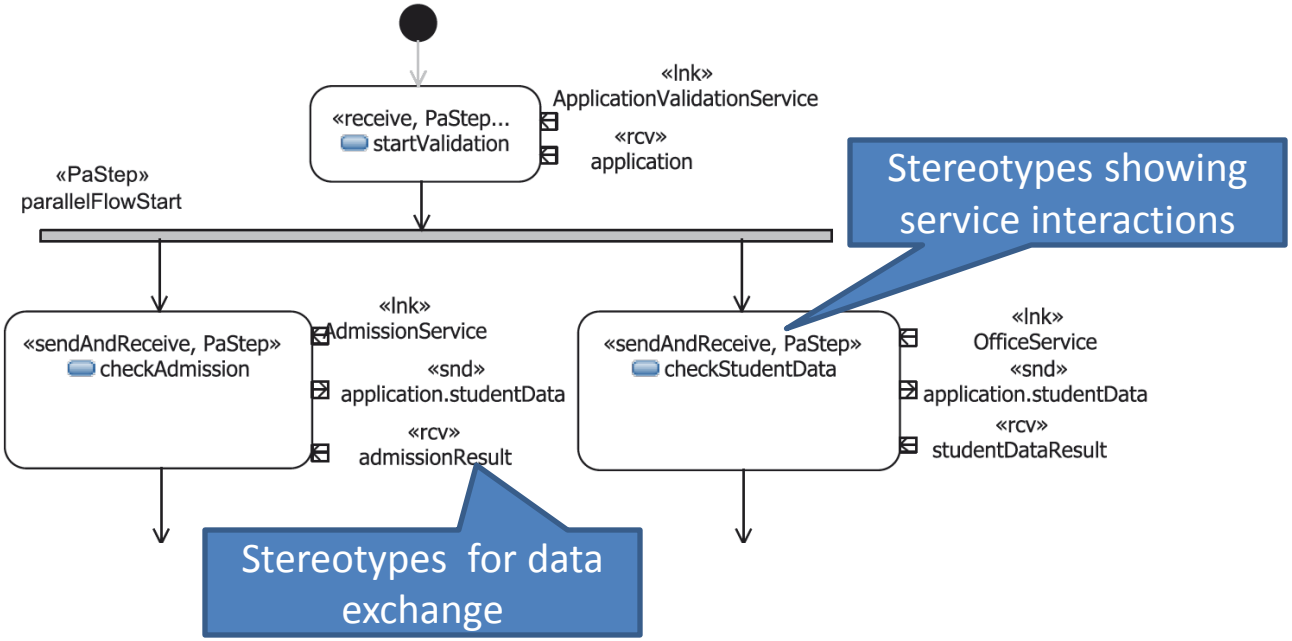
www.sensoria-ist.eu



Orchestration

Orchestration

- The two orchestrations are modelled in UML4SOA
- They communicate with each other to exchange data about the student
- Example:



- The non-functional aspects used in the eUniversity student registration scenario will be detailed in the following.

Non-functional properties of the case study

- The Client and the ApplicationCreationService should communicate via a **secure and reliable connection.**
- The document UploadService might be under **heavy workload**, therefore its throughput should be at least X requests/second with a Y second average response time.
- All **requests** sent to the ApplicationValidationService **should be acknowledged.**
- As the validation service handles **confidential** data, all requests should be **encrypted** in order to protect the privacy of the students.
- Messages sent by the ApplicationValidationService must be clearly **accountable**, i.e. the decisions must be **non-repudiation** of messages must be guaranteed.

What is now validated?

- „Completeness and correctness” of the model
- Is there any contract in the model?
- All contract should have at least one characteristic (NF aspect) defined
- All characteristics should have at least one dimension (group of attributes) defined
- All definitions should be unambigous

What else could be validated?

- Requirements which are specific to a concrete aspect
  - E.g. dependability: timing parameters should be defined
- Domain specific requirements
  - e.g. in a banking process, are SLAs defined for all participants?
  - Are all connections secured by design?
- „Best practice”
  - All bodies should be encrypted but none of the message headers
  - The entire message should be signed
- Middleware-specifi parameters
  - E.g. values should be within a given range

Some technicalities....

- Always export your UML project as UML2 (EMF) files
- Always delete the VIATRA representation before parsing the model again
- Always read the error log and the VIATRA textual output…
- Order of parsing the transformations
  - Validator
  - XML (first the general, then the specific ones)
  - umlrampart, umlsandesha (then one calling these)

Homework

- Finish the SAL template and the concrete SLA
- Generate the pseudo-code
- Extend the validator transformation with at least two patterns
  - One „general”, one „aspect-specific”