Turning a Suite of Modeling and Processing Tools Into a Production Grade System

Caisse nationale d'assurance vieillesse (Cnav)
110-112, avenue de Flandre 75019 Paris
• **Pascal Rivière** Head of Methodology Dept • pascal.riviere@cnav.fr
• **Olivier Rosec** Head of Data Interchange Unit • olivier.rosec@cnav.fr

The French national agency for old age pensions has evolved, along a systemic vision of all the issues to be covered, a generic suite of tools to design and process structured file formats. This suite of tools relies for its distribution on continuous integration, seen here as a system of systems.

A Suite of Tools for the Whole Community
(Emitters as well as Recipients of Structured File Formats)

The generic suite of tools covers the following functionalities (corresponding tool between brackets):

- Modeling an interchange format and defining its validating rules (Designer). This tool is used by the Interchange Standard Technical Committee.
- Generating the specification of the interchange format (Designer). These deliverables target the implementers of the file format in any software product which must be made compliant with it.
- Generating validating components which can be integrated in a process (Validating building block = Validator + Knowledge Base). This component is intended for the integration team in charge of the platform which must validate files complying with the standard.
- Generating a validating tool with a graphical user interface for a local compliance check before sending files to the remote processing platform (Auto Validator). This tool targets end users who emit the files carrying the business data.

A System of Systems: Continuous Integration Serving a Release Plan based on {Tool Version; File Format Version} Couples.

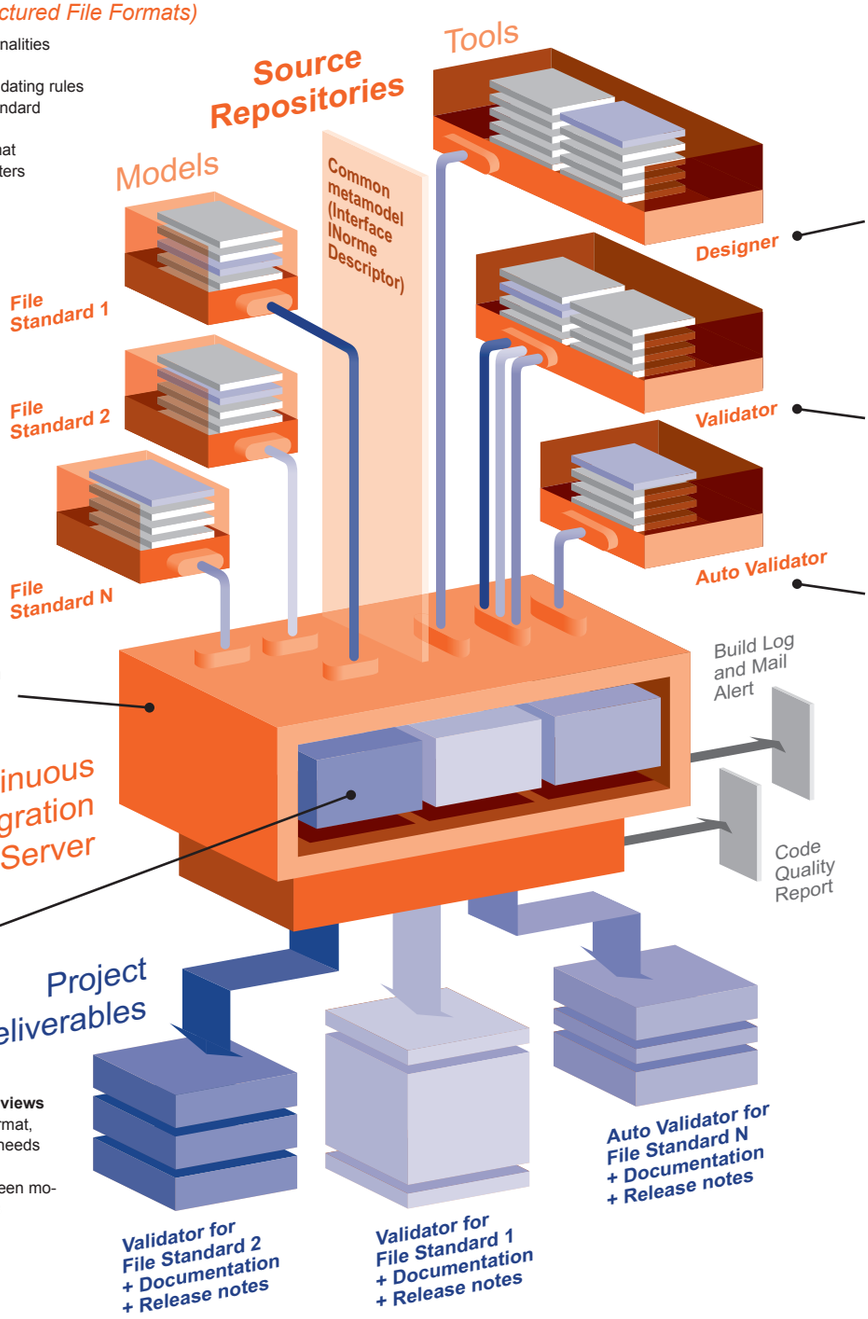
Continuous integration has here turned into a system which industrializes the production of suites of tools, each characterized by the file format it implements and the version under which it is released according to the shipping schedule of the project.

Typical Building Jobs

Different products are assembled from shared building blocks. For example:

- **Building a Validator** means assembling a **validator** (engine) and a **knowledge base** (file format specification read by the engine as it checks files against the specification before they are fed into a business database)
- **Building an Auto Validator** means assembling a **GUI** and its **extension points** on top of a **validator**. Among the extension points are the **knowledge base**, the **file format and data views** and **report stream structures** particular to a file format, and the **branding of the product** according to the needs of the project (titlebars, icons, etc.)

Genericity still prevails thanks to the separation between models representing business rules and transformations implementing the processing logic.



The Suite of Modeling and Processing Tools as a Line of Products

The **Designer** *is an IDE with a specific perspective for modeling file formats and scripting validating rules. It generates:*
• The ***specification of the file format*** *which will be disseminated throughout the user community for implementation (.pdf, .csv exports, XML schemas)*
• The ***knowledge base*** *which will be read by the Validator to check whether the actual files sent by the users comply with the specification*

The **Validator** *goes through several stages to process a file:*
• ***Conversion into XML***
• ***Syntactical validation***
• ***Semantic validation***
As the file is processed, the Validator emits a report stream which can be customized to the needs of any user community, thanks to an architecture which separates events logging and report serialization

The **Auto Validator** *is equivalent to a Validator with a GUI. It can be distributed to the users so they can check locally whether their files comply with the specification before sending them to the processing platform.*
The Auto Validator tool reads the same knowledge base and executes the same validating logic as the Validator rolled out in production.

A **testing tool** ***separate from the IDE*** *can be distributed to qualifying teams* ***to validate the initial knowledge base*** *against the specification it implements and* ***to ensure non-regression on validating components*** *before roll-out. The Tester loads a series of test cases which are then automatically executed and parsed to compare the obtained result with the expected result.*

Implementation
Java, Eclipse Modeling Framework, Jenkins, Maven, Tycho...

CSD&M 2014

Conception graphique et infographie : **Tony Gonçalves** *techoppe.tony@wanadoo.fr*