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

Pascal Rivière¹, Olivier Rosec¹

 Caisse nationale d'assurance vieillesse (Cnav)
 110 – 112 avenue de Flandre, F-75019 Paris pascal.riviere@cnav.fr, olivier.rosec@cnav.fr

Abstract. The French national agency for old age pensions has evolved, along a systemic vision of all the issues to be covered, a generic set of tools to design and process structured file formats. This set of tools has turned into a production grade modeling and validating suite which relies for its distribution on the continuous integration of its various components. As such, continuous integration has here turned into a system which industrializes the production of systems of tools, each characterized by the file format it implements and the version under which it is released.

1 Introduction

Social security agencies have to rely on a massive influx of data to assess benefits and collect contributions. The data comes in thanks to structured file formats. Processing and validating these files is not easy, because of the many changes in their specification each year.

This is why the national agency for old age pensions (Cnav) wanted to avoid rewriting endlessly its file processing applications, and wanted to maintain a constant level of processing quality. A general approach was necessary.

To achieve this, the national agency has elaborated a generic set of tools [Rivière, Rosec 2013]. It was initially used on the N4DS standard, and then on the new DSN standard (N4DS = Norme de Déclarations Dématérialisées De Données Sociales, DSN = Déclaration Sociale Nominative).

This approach can be applied well beyond the initial problem domain: collecting data from payroll systems.

The suite of tools has been improved and extended along a systemic view of all issues to be covered when dealing with structured file formats: design, validation logic, testing logic, documentation both of the file formats and the semantics of the business data they carry.

2 What is an interchange standard?

Elaborating a generic set of tools for modeling interchange formats and for generating the corresponding validating components brings back to the original issue: defining a metamodel for an interchange standard. We define it by the following characteristics:

- It should implement a conceptual model of the business domain;
- It can be derived into a series of message models carrying the business data payload
- Each message is a hierarchy of data blocks which respect a certain structure and multiplicity (0,*)
- The structure of a block is defined as an ordered series of data elements which respect a certain syntax and multiplicity (0,1)
- The value and syntax of a data element is defined by a business datatype restricting one of four technical datatypes: string, decimal, enumeration, external referential
- Each technical datatype has facets which help define business datatypes: minimum and maximum length, a regular expression for string and decimal etc.
- Semantic consistency between data elements is enforced by rules.
- Because the interchange standard has to support the interchange scenarios defined by its user community, message models can themselves be inserted between a header and a footer composed of blocks linked to the applicative logic of the actual exchange system.

On top of this metamodel one can model many interchange formats. Here are the metrics for a few of the interchange formats modeled with the suite of tools.

Number of	N4DS v01x08	DSN phase 1	DSN phase 2
Data blocks	124	29	33
Data elements	687	202	258
Validating rules	995	198	210

The DSN message is the first for which a conceptual model was elaborated before working on the corresponding file formats. The idea is to derive the message model from the class diagram (below). It should be noted that such a derivation cannot be 100% deterministic and requires some choices [Elmasri, Navathe 2011].

Entreprise	1				action sociale complémentaire collective
Mith D.C. KCouvepr D.T.	Arrit de travail	Autre suspension de l'exécution du contra	Fin de con		de l'adhésise de protection aussise complémentaire (B. 1) ataire de gestion (B. 1)
(a 4.4% ; S. 1) denotes of ball (b a color (), 1) denotes (C ball) denotes of ball (b a color (), 1) denotes (), 2) denotes (), 3, 1) den	BO: (0, 1) Doi: on other dis subregation: (0, 1) Doi: on other dis subregation: (0, 1) Doi: on the origination: (0, 1) Doi: on the origination: (0, 1) Doi: on a restore (0, 1) Doi: on the restore (0, 1)	(2) Den an debit dis subjects. (5, 4) (3) Den ante, (5, 5) (3) Met An subjects (5, 5) (3) (4) (5) (5) (5) (5) (5) (5) (5) (5	Des et les avants (B, 1) Des et les avants (B, 1) Des des et les avants (B, 1) Des departs (B, 2) Des departs	Concenser: 2.1 Marker: 2.1 Marker: 2.1 Marker: 2.1 Native: 3.1 1 a same to const CD ² , 3.1 1 a same to const CD ² , 3.1	tion de protectur salais complementais - 20. 1
Endudu			CB Basire nel fonsire du calarié (E. 1) CB Statut perficuler du calarié (E. 1)		And the second s
Lill Number dimoniphen aur Hamilten (d. 1). Kill Bener (d. 1)	E Lies	fe toval	TT (di Telescorences) (S.C)		1.1.
III favo perfamilie (D. 1) III Nov ducage (D. 1)	Constant allow default (2) Constant (2) (3)	g 70 rs	Contrat (contrat de travail, e	envention, mandati II	Code Higher Battale complementarie
El la la matanta (E) (El la la matanta (E) (El la la matanta (E) (El la la la matanta (E) (Cone de statuture à titranger Cone Reps (21.1) Cone Reps (21.1)	77	$\label{eq:constraints} \begin{array}{c} (1) does a straints of the straints of th$		Munches and a complementaria field with the maximum of a first set of the se
xxxXX (0.1) organization (0.1) (0.1)		Rémuné	ration	Prime, gratification et indomnité	Gil Kentfant Drigeriere de Armeter Sociale (D. 1
ence de las tables conditions de la calculation de la cal- la de la foi de la calculation de la calcul	-	ID Total (R, IL) ID Marker (R, IL) ID Totalse (R, IL)		a Yor, B, I De in Alba in alcos in statuteer, B Herart, D, I De in the prime in statuteer, B, I De inter by prime in statuteer, B, I Autor diamet de revenu bot	Different Faculative, B. 1 Different die contraction, B. 1 Different die contraction, B. 1 Different die contraction, B. 1 71
7 p			10 ACSV18 / 17 (2	2 Type (8.1) 2 Date de clánal de pérsoin de reflectement (8.1)	
	au de cotisation due		El Meure (E.I)	Startant (D. 1) Date in fit de période de reflectement (D. 1)	
té d'affaitation des quérations (D. 1)		art in consider (B.1)	GL Grab de resource (B. 1)		El Rese assujettia
1 M M		P. D.	17		CL Meter de las assuites (E. 1)

Fig. 1. DSN phase 2 class diagram

The figure below shows how various message models can be inserted between header and footer blocks.



Fig. 2. DSN phase 2 Message Hierarchy

The figure below shows how the biggest DSN message model, the monthly DSN, is represented as a hierarchy of data blocks in the Designer.



Fig. 3. DSN phase 2 Monthly Declaration Hierarchy

Each data element is described in detail in the .pdf export of the specification.



3 Tools, what for?

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

- Modeling an interchange format and defining its validating rules (Designer)
- Documenting the data carried by the interchange format (Semantic Repository)
- Generating the specification of the interchange format as a .pdf + various .csv ex-ports (Designer)
- Generating validating components which can be integrated in a process (Parser within the Designer, Validating building block composed of Knowledge Base cum Validator)

- Generating a validating tool with a graphical user interface for a local compliance check before sending files to the remote processing platform (Autovalidator)
- Testing the generated components (Tester)

This series of tools does not owe its birth to a careful plan. One should rather think of it as an emergence. It is through the efforts of the national agency to organize the original system of goals and means and the various artefacts supporting them into a coherent whole that new tools kept arising until the present list was obtained.

How do these tools operate? This is what this part of the paper is about.

3.1 The Designer

This tool is at the heart of the suite. As previously noted, the Designer generates:

- The specification of the interchange format which will be disseminated throughout the user community for implementation (.pdf, .csv export, XML schemas)
- But also the knowledge base which will be read by the Validator to check whether the actual files sent by the users comply with the specification

As a modeler of data structures, datatypes and messages, the Designer is fairly classical. But as a repository of validating rules relying on first-order predicate logic, it is not.



Fig. 5. Top part of the Designer UI

In the modeling perspective, the screen is split up between the project view (top left), the outline view of the current model (bottom left), the view displaying the current model element (top right) and the view showing the properties of the part of the model currently in focus (bottom right).

< m		Properties 🛙 🔐 Problems 5	Impact Analysis 🤨 Error Log 👘	- 0
🗄 Outline 🕮	A 10 🖸		[1] > 데 ID 전	. 7
💷 Bundle DSN	•	Property	Value	
[Data Types DSN		Allowed		
Structures DSN		Local Usage In Declaration	Mandatory	
DSN S20.G00.05 - DSN Mensuelle		Rules	+ CRE-11 : is_in_file(Srub;@REF_NAF)	1
S20.G00.05 - DSN SIGNAL ARRET TRAVAIL		Values		
S20.600.05 - DSN SIGNAL ARRET TRAVAL		Data Type Identification R/O		
in S20.000.05 - DSN SIGNAL REPRISE TRAVAIL	1	Description	IT INSEE /NAF revision 2	
Data Sources		ld	Insee_Naf_2008_4	
Na Documents		Data Type Misc R/O		
A Recipients		Comment	12	
Rules Configuration		Max Length	Fi 5	
Aliases	1.11	Min Length	F4 5	
Q Macros		Nature	12 ExternalReferential	
2. External Europtions	-	(*)	10	
5º Selected Object: 003 - Code APEN				

Fig. 5. Bottom part of the Designer UI

Rules are modeled under the data element to which they are attached in the functional specification for the file format.

CCH-1	éro du contrat de travail 1	
and the second s	- 2 : every x:S21.G00.40,y:S21.G00.40 satisfies (((\$x!=\$y) and is_present(\$x:S21./	
📫 eve	ry x:S21.G00.40,y:S21.G00.40 satisfies (((Sxl=Sy) and is_present(Sx:S21.G00.40	
💊 010 - Date	de fin prévisionnelle du contrat de travail	
💊 011 - Unité	de mesure de la quotité de travail	
\$ 012 - Quot	ité de travail de référence de l'entreprise pour la catégorie de salarié	
S 013 - Quot	ité de travail du contrat de travail	
\$ 014 - Mod	alité d'exercice du temps de travail	
💊 015 - Salais	e de référence porté par le contrat	
💊 016 - Régin	ne local Alsace Moselle	
🖾 🔠 Proble	ems 😼 Impact Analysis 🕙 Error Log	1 × 1 × 1 × 1 × 1 × 1 × 1 × 1 × 1 × 1 ×
🖾 📳 Proble	ms 💆 Impact Analysis 🥂 Error Log Value	11 × 2 × 10
		∃ ≯¤⊨# [▼] °C
xt	Value	E > a < E <
xt el	Value S21.600.30 - Salarié	
xt al f Values	Value S21.600.30 - Salarié	E > a a a a a a a a a a a a a a a a a a
xt al d Values ige	Value S21.600.30 - Salarié Ma false	■ > ■ ■ ■ ■
i 🖾 👬 Proble ext ral d Values ege ct tion	Value S21.600.30 - Salarié Ma false	

Fig. 5. Rule as displayed in the Structures and Properties views

A rule has a subject and a context, that is, a position within the message tree from which all paths to the data elements called by the rule will be calculated. The textual DSL corresponding to the functional rule is entered in the model through a widget.



Fig. 5. Rule as displayed in the DSL widget

A message will be returned to the user in the report stream if the processed file does not satisfy the rule.



Fig. 5. Rule user message

When the knowledge base is compiled, the rule will be implemented as a Java class along with the set of utility classes loading and unloading at runtime the data elements called by the rule.

Norm - th chev norme dan p01v01/arc-geey/th/chev/horme/dan/p01v01/vu	
File Edit Source Refactor Navigate Search Project Run Winds	× Μφ ₹ {} + 3 + 5 Φ + 0 +
21 G. Gt Repository Deploring Mg Norm	
Sth Norm Explorer 11 2 2 2 2 2 2 2 2	N405, bundle.nem 1/ Rule521, 000, 43, 009, CCH 32, pm 1
April 2010, 2011, 2011, 2011, 2011, 2011 April 2010, 2011, 20	<pre>} IBulsessit rulsesuit = ruli; Bocien ruuit = ruli; /ruli = reprivat_ment = ruli; perivat_ment = ruli = ruli</pre>
20 Outries 12 ED Outries 12 ED Outries 12 ED Outries 12 21 month disclaritions Impact disclaritions Impact disclaritions Impact disclaritions 2 ¹¹ Disclaritions Impact disclaritions Impact disclaritions Impact disclaritions 31 month disclaritions Impact disclaritions Impact disclaritions Impact disclaritions 32 month disclaritions Impact disclaritions Impact disclaritions Impact disclaritions 32 month disclaritions Impact disclaritions Impact disclaritions Impact disclaritions 34 month disclaritions Impact disclaritions Impact disclaritions Impact disclaritions 34 month disclaritions Impact disclaritions Impact disclaritions Impact disclaritions 35 month disclaritions Impact disclaritions Impact disclaritions Impact disclaritions 36 month disclaritions Impact disclaritions Impact disclaritions Impact disclaritions 36 month disclaritions Impact disclaritions Impact disclaritions Impact disclaritions 36 month disclaritions Impact disclaritions Impact dimpact dimpact dimpact disclaritions	<pre></pre>
g _ run(SourGroupe) BulePeoult	Departies 11 🛃 Problems 🗇 Impact Analysis 👻 Erner Log

Fig. 5. Java implementation of a rule

3.2 The Validator

When a file is processed by the Validator, it goes through several stages:

- Conversion into XML
- Syntactical validation
- Semantic validation

As the file is processed, the Validator emits a report stream. The screenshot below shows the console and part of the report as transformed from XML to HTML by a local script.

14-04-12 11:05:30,507 # 1 14-04-12 11:05:30,507 # 1 14-04-12 11:05:30,507 # 1 14-04-12 11:05:30,507 # 1	e the lo ng.apach NFO = N4 NFO = N4 NFO = N4 NFO = N4	z.org/log4j⁄1.2/faq.html# DSShell # Opúration rúali DSShell # Rüsultat du tra DSShell # code retour d DSShell # Rúsultat du tra DSShell # code retour d	súe en : 468ms . itement : u service : 6 itement :	Google .	P D •
NDocuments\Environment\Ie	st∖MH_vi				
NDocuments\Environment\Te	st∖MH_v1			niveau er	nvoi
Contexte	Ligne	Controle	Rubrique	niveau er Valeur	nvoi Salari
	Ligne	Controle CST-02	Rubrique \$10.G00.95.001	Valeur test	Salari
Contexte 19966_FichierExport.dsn type	Ligne 23	CST-02	\$10.G00.95.001	Valeur test niveau décla	Salar
Contexte 19966_FichierExport.dsn/type Contexte	Ligne 23 Ligne	CST-02 Controle	S10.G00.95.001 Rubrique	Valeur test	Salar
Contexte 19966 FichierExport.dsn type Contexte rang 1 nature type	Ligne 23 Ligne 0	CST-02 Controle CSL-00	\$10.G00.95.001 Rubrique \$20.G00.96.902	Valeur test niveau décla	Salar
Contexte 19966 FichierExport.dsn type Contexte rang1 nature type rang1 nature type	Ligne 23 Ligne 0 0	CST-02 Controle CSL-00 CST-02	S10.G00.95.001 Rubrique S20.G00.96.902 S20.G00.96.902	Valeur test niveau décl: Valeur 1 1	Salar aration Salar
Contexte 19966 FichierExport.dsn type Contexte rang 1 nature type	Ligne 23 Ligne 0	CST-02 Controle CSL-00	\$10.G00.95.001 Rubrique \$20.G00.96.902	Valeur test niveau décla	Salari aration Salari SN

Fig. 5. Command line view of the Validator against backdrop of HTMLized report

The Validator has been improved: the report stream can be customized to the needs of any user community, thanks to an architecture which now separates events logging and report serialization

The Validator has been rolled out in production for the DSN.

A POC has been made to validate file formats circulating between old age pensions agencies to consolidate data about pensioners claiming the minimum entitlement.

3.3 The Tester

Work is under way on a testing tool which should:

- Validate the initial knowledge base against the specification it implements
- Ensure non-regression on validating components before roll-out
- Generate semi-automatically test suites.

The Tester is an RCP tool which loads a test suite file into a series of test cases which are then automatically executed and parsed to compare the obtained result with the expected result.

3.4 The Auto Validator

Users can now check whether their files comply with the specification before sending them to the processing platform. The Autovalidator tool reads the same knowledge base and executes the same validating logic as the Validator in production.



Fig. 5. Screenshot showing local validation with DSNVAL tool

Anyone can download the Autovalidator tool for the DSN, DSNVAL, on the Internet: <u>http://www.dsn-info.fr/precontrole-dsn-val.htm</u>

3.3 The Semantic Repository

A Semantic Repository now links data definitions coming from the conceptual data model of the business domain to their implementation in the file format specification. A query tool helps navigate through the Repository, using various approaches: selection criteria, full text research, keywords...



Fig. 5. Welcome page of DSN Semantic Repository

4 Implementation.

All the tools call on the facilities offered by the Eclipse Modeling Framework.

The Designer is built on top of the Eclipse platform. An effort has been made to organize its plug-ins into coherent features. It has moved from a mere graphical modeler to a developing tool which federates into a specialized plug-in all the projects an interchange format relies on for its successful implementation: modeling projects, and Java projects too in the guise of the functions extending the textual DSL, the few remaining rules written directly in Java, the Java serialization for the report stream.

A class known as a Norm Descriptor calls the validating logic persisted in the knowledge base. The knowledge base itself is a Java archive organized in layers according to the processing stages of the Validator.

In the models layer, one finds the files which describe the syntax of the model the actual files have to comply with, in terms of data blocks, elements and types.

A Java project carries the logic for the semantic validating rules and the report serialization.

Report logic relies on a model which is shared by the Designer and Validator.

The conversion algorithm which transforms the input files into XML has evolved at the beginning of 2014 to throw an error when a data block or element is out of sequence with regard to the "covering message" (the superset of all message models for a file standard). It thus contributes to the validating logic. Previously it generated extra data blocks elements to generate a sequence of XML elements which would always be valid, but this strategy led to misleading error reports at the syntactical stage.

This is the most recent change brought about by the need to bring the Validator to production grade level, after the replacement of the initial off-the-shelf engines. The semantic validation was ported from Saxon and Schematron to a Java API in 2012 for performance, and the syntactical validation was ported from Xerces to the same Java API because users did not understand report messages phrased in XML constructs.

5 Continuous integration.

Originally, continuous integration covered only the suite of tools which were then more or less used on a standalone basis by the unit which specializes in the generation of the format models and validating components within the national agency.

But as the Validator was being prepared for production, it was considered safer to include in the testing strategy the knowledge base itself.

Now continuous integration encompasses all source control repositories, whether for tools or file formats, and builds to roll out fully tested components can be specialized according to the phase and version of the file format for which deliverables must be shipped.



Fig. 5. Code repositories called by builds according to tools and deliverables

Continuous integration has become a system which helps produce systems of tools for structured file formats, each system of tools and deliverables being specialized into a given file format and released under a certain version.

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

One could almost describe the system of code repositories and build scripts as an autopoietic machine perpetually regenerating tools and deliverables.

6 Conclusion

The suite of tools is used intensively and successfully for the DSN project.

The national agency wants to make it yet again more generic in order to generalize its use for the validation of most or all structured data interchanges between other institutions and itself.

It is also investigating the possibility of turning the suite of tools open source in the hope it nurtures a new way of thinking about and working with structured file formats.

References

1. Pascal Rivière, Olivier Rosec, "Model-Based Interchange Formats: a Generic set of tools for Validating Structured Data against a Knowledge Base", CSDM 2013

2. R. Elmasri., S.B. Navathe, "Fundamentals of database systems", Addison-Wesley 2011