Message Structures: a modelling technique for information systems analysis and design[1]

Arturo González[1], Marcela Ruiz[2], Sergio España[2], Óscar Pastor[2]

[1]Departamento de Sistemas Informáticos y Computación
[2]Centro de Investigación en Métodos de Producción de Software (ProS)
[1,2]Universidad Politécnica de Valencia, España
agdelrio@dsic.upv.es
{lruiz, sergio.espana, opastor}@pros.upv.es

Abstract. Despite the increasing maturity of model-driven development (MDD), some research challenges remain open in the field of information systems (IS). For instance, there is a need to improve modelling techniques so that they cover several development stages and they facilitate the transition from analysis to design. This paper presents Message Structures, a technique for the specification of communicative interactions between the IS and its environment. This technique can be used both in analysis and in design. During analysis, it allows abstracting from the technology that will support the IS, and to complement business process modelling with the specification of communicational needs. During design, Message Structures serves two purposes: (i) it allows to systematically derive a model of the IS memory (e.g. a UML class diagram), (ii) and it allows to reason the user interface design using abstract patterns. The technique is part of Communication Analysis, a communication-oriented requirements engineering method, but it can also be used in combination with other modelling techniques (e.g. Business Process Modeling Notation, Use Cases). Two supporting tools are presented: one uses the Xtext technology, and the other uses the Eclipse Modelling Framework.

Keywords: Message Structures, Communication Analysis, analysis, design, information systems, requirements engineering, model-driven development.

1 Introduction

Despite the increasing maturity of model-driven development (MDD), some research challenges remain open in the field of information systems (e.g. to facilitate business process modelling from a communicational perspective [1], to provide model transformations from requirements to design, to improve requirements traceability).

[1] Research supported by the Spanish Ministry of Science and Innovation project PROS-Req (TIN2010-19130-C02-02), the Generalitat Valenciana project ORCA (PROMETEO/2009/015), the Spanish Ministry of Education FPU grant (AP2006-02323), and co-financed with Structural Funds from the European Regional Development Fund.

This paper presents a technique for the specification of communication with the information system (IS): Message Structures[2]. Although it is part of Communication Analysis, a communication-oriented requirements engineering method [1], it can be used in combination with other methods and notations as well (e.g. Use Cases, Business Process Modeling Notation). Also, Message Structures can be applied solely in analysis time, solely in design time, or throughout the entire lifecycle.

Previous works present Message Structures concisely [1-3]. This paper extends these works and makes the following contributions:

- A precise specification of the technique is provided, including definitions and examples of its concepts, its grammatical constructs, and its acquisition operations.
- Different ways if using the message structures are described, differentiating its application in analysis time (specification of business processes from a communicational perspective) and its application in design time (derivation of IS memory models and reasoning of the user interface in terms of abstract patterns).
- Two alternative modelling tools for Message Structures are presented; one is based in Xtext and the other is based in Eclipse Modeling Framework.

The rest of the paper is structured as follows. Section 2 presents an overview of Communication Analysis. Section 3 presents in detail the concepts related to Message Structures, and describes its grammar. Section 4 differentiates its application in analysis time and in design time. Section 5 presents the tool support. Section 6 reviews related work. Section 7 discusses some advantages and risks of using Message Structures and presents future work.

2 Overview of Communication Analysis

Communication Analysis is a requirements engineering method that proposes describing business processes from a communicational perspective. The method stems from academic research and it evolves in collaboration with industry [2]. The following definitions clarify some conceptual foundations of the method.

We refer as *communicative interaction* to an interaction between actors with the purpose of exchanging information. Depending on the preeminent direction of the information, two types of communicative interaction are distinguished:

- *Ingoing communicative interaction*. Its main objective is to convey to the system new meaningful information; i.e. to feed the IS memory with previously unknown data that is relevant to the organisation.
- *Outgoing communicative interaction*. Its main objective is to distribute known information to users; i.e. to retrieve data from the IS memory and to present it.

Industrial experience has shown us that ingoing communicative interactions entail more analytical complexity. We recommend focusing IS analysis on them.

[2] This technique has been named differently during the evolution of Communication Analysis: Data Acquisition Structures [2-3], Communication Structures [1]. We consider that the term Message Structures reflects their essence more appropriately.

A *communicative event* is a set of actions related to information (acquisition, storage, processing, retrieval, and/or distribution), that are carried out in a complete and uninterrupted way, on the occasion of an external stimulus [2]. For Communication Analysis, a communicative event is an ingoing communicative interaction that fulfils certain unity criteria (i.e. guidelines that facilitate the creation of modular models) [4]. Informally, it can be seen as a business process activity.

Communication Analysis proposes specifying business processes by means of Communicative Event Diagrams, a notation which is similar to UML Activity Diagrams. Event Specification Templates is a textual specification technique that prescribes a structure for the requirements related to a communicative event. Previous works describe these techniques in detail [1]. In the following, we focus on a technique for the specification of messages associated to business process activities.

3 Message Structures

Message Structures is a specification technique that allows describing, by means of structured text, the message that is associated to a communicative interaction. Although message structures can be used to specify outgoing communicative interactions, due to space limitations we focus on the specification of ingoing communicative interactions, given their analytical interest.

Table 1. Example of a message structure in analysis time[3]

FIELD	OP	DOMAIN	EXAMPLE VALUE
ORDER =			
< Order number +	g	number	10352
Request date +	i	date	31-08-2009
Payment type +	i	text	Cash
Client +	i	Client	56746163-R, John Papiro Jr.
DESTINATIONS =			
{ DESTINATION =			
< Address +	i	Client address	Blvd. Blue mountain, 35-14A, 2363 Toontown
Person in charge +	i	text	Brayden Hitchcock
LINES =			
{ LINE =			
< Product +	i	Product	ST39455, Rounded scissors (cebra) box-100
Price +	i	money	25,40 €
Quantity >	i	number	35
}			
>			
}			
>			

Table 1 presents an example of the usage in analysis time; the ORDER message structure specifies a communicative interaction by which a client places an order (the example is taken from a requirements model that can be found in [6]).

[3] This particular font, the colours and the capitalisation are a non-prescriptive convention that is intended to facilitate message structure comprehension. Feel free to configure these aspects.

3.1 Grammatical constructs

In the following, the syntax of Message Structures is described. Due to space limitations, the formal grammar rules can be found in [5].

We refer as *substructure* to an element that is part of a message structure. This way, LINE, Client and Payment type are substructures that are part of ORDER. There exist two classes of substructures: fields and complex substructures. We refer as *initial substructure* to the substructure that constitutes the first level of a message structure. For instance, ORDER=<Order number+Request date+Payment type+Client+DESTINATIONS>.

- *Field*. It is a basic informational element of the message (i.e. it is not composed of other elements). There exist two types of fields.
  - *Data field.* It is a field that represents a piece of data with a basic domain (basic domains such as numbers and text are discussed below). For instance, Payment type is a data field that belongs to the message structure ORDER.
  - *Reference field.* It is a field whose domain is a type of business objects. For instance, Client references a client that is already known by the IS.
- *Complex substructure*. It is any substructure that has an internal composition.
  - *Aggregation substructure.* It specifies a composition of several substructures in a way that they are grouped as a whole. It is represented by angle brackets < >. For instance, LINE=<Product+Price+Quantity> specifies that an order line consists of info about a product, its price and the quantity requested by the client.
  - *Iteration substructure.* It specifies a set or repetition of the substructures it contains. It is represented by curly brackets { }. For instance, an order can have several destinations and, for each destination, a set of order lines is defined. Both DESTINATIONS and LINES are iteration substructures.
    LINES={LINE=<Product+Price+Quantity>}
  - *Specialisation substructure.* It specifies one or more variants (i.e. structural alternatives). There is no example of specialisation substructure in Table 1. Given MSG=<a+b+OPTION=[c|d]+e>, both <a+b+c+e> and <a+b+d+e> are valid messages; see [5] for a more illustrative example.

3.2 Field specification

To characterise a field, the following properties can be specified:

- *Name*. Each field must have a significant name (e.g. Request date).
- *Acquisition operation*. It specifies the origin of the information that the field represents.
  - *Input* i. The information of the field is provided by the primary actor.
  - *Generation* g. The IS can automatically generate the information of the field.
  - *Derivation* d. The information of the field is already known by the IS and, therefore, it can be derived from its memory; that is, it was previously communicated in a preceding communicative event. This operation can have an associated derivation formula.
- *Domain*. It specifies the type of information the field contains.

- *Example.* An example of a value for the field, provided by the organisation.
- *Description.* An explanation that helps the reader to understand the field meaning.
- *Label*. A brief text that describes the field when shown in a graphical interface.
- *Link with memory*. It specifies the correspondence between the field and a database table column or a class diagram attribute.
- *Compulsoriness*. It specifies whether the message field is mandatory or not.
- *Initialisation*. The value that the field is given by default can be specified by means of a function or a derivation formula.
- *Visibility*. It specifies whether the field is visible in a graphical user interface form.

It is recommended to lay the fields out vertically and to specify the field properties horizontally (by means of columns). For reasons of space, the description of the fields can be done in a separate table. Message Structures can be extended with other field properties that a method designer or an analyst deem appropriate. However, as discussed below, not all properties are convenient at analysis time.

4 Usages of Message Structures

Message Structures can be applied for different purposes (from software development to adaptive maintenance) and in different stages of the software development life cycle (e.g. analysis, design). Depending on whether they are used in analysis or design time, syntactic and pragmatic differences have to be taken into account. Table 2 presents recommendations on the usage of field properties, depending on the development stage in which Message Structures are used.

Table 2. Applicability of field properties to development stage

		Name	Acquisition operation			Domain	Example	Description	Label	Link with memory	Compulsoriness	Initialisation	Visibility
			i	g	d								
Analysis		++	++	++	--	++	++	++	--	--	--	--	--
Design	Memory	++	++	++	++	++	++	++	-	++	+	-	-
	Interface	++	++	++	++	++	++	++	++	++	++	++	+

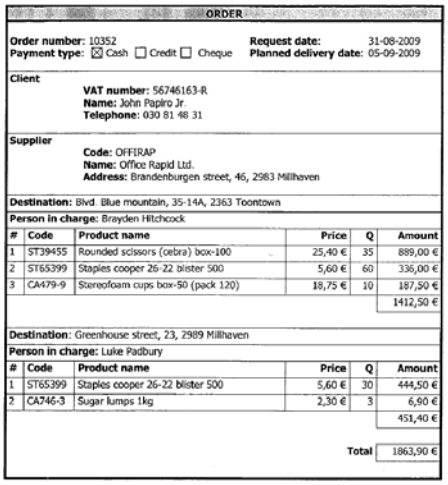
LEGEND: ++ highly recommended + recommended - not recommended -- discouraged

4.1 Creation and usage of Message Structures in analysis time

In analysis time, Message Structures allow specifying in detail the communicative interactions that take place in the organisational work practice. This way, they offer a communicational perspective for business process modelling and they act as requirements for the IS. In the context of Communication Analysis, the new meaningful information that is conveyed to the IS in each communicative event is

specified by means of a message structure. In the following, we enumerate some valuable sources of information and techniques for analysing IS communications.

Organisational actors play an important role in IS analysis, since they know organisational work practice first-hand. The analyst will interview them.



ORDER

Order number: 10352 **Request date:** 31-08-2009
Payment type: ☒ Cash ☐ Credit ☐ Cheque **Planned delivery date:** 05-09-2009

Client
VAT number: 56746163-R
Name: John Papiro Jr.
Telephone: 030 81 48 31

Supplier
Code: OFFIRAP
Name: Office Rapid Ltd.
Address: Brandenburgen street, 46, 2983 Millhaven

Destination: Blvd. Blue mountain, 35-14A, 2363 Toontown
Person in charge: Brayden Hitchcock

#	Code	Product name	Price	Q	Amount
1	ST39455	Rounded scissors (cebra) box-100	25,40 €	35	889,00 €
2	ST65399	Staples cooper 26-22 blister 500	5,60 €	60	336,00 €
3	CA479-9	Stereofoam cups box-50 (pack 120)	18,75 €	10	187,50 €
					1412,50 €

Destination: Greenhouse street, 23, 2989 Millhaven
Person in charge: Luke Padbury

#	Code	Product name	Price	Q	Amount
1	ST65399	Staples cooper 26-22 blister 500	5,60 €	30	444,50 €
2	CA746-3	Sugar lumps 1kg	2,30 €	3	6,90 €
					451,40 €

Total 1863,90 €

Fig. 1. Example of an order form

Business forms are a technological support for communicative interactions and, therefore, they are a major source for analysis. In this sense, the user interface screens from pre-existing software are equivalent. In analysis time, input forms allow to identify communicative events that convey new information to the IS. The analyst has to carry on, along with the users, the following investigations:

- Whether the form is filled in one go or iteratively in different moments in time; the corresponding communicative events are identified. For instance, the client order form shown in Fig. 1 is affected by more than one communicative event [6]: the request of the order, the assignment to a supplier, the supplier response.
- Which is the temporal order in which communicative events occur.
- Which are the primary actors of the communicative events (those that are the source of the information the form is filled which).
- What message is conveyed in each communicative event (the fields of the form that are affected by the event). Observe that the message structure in Table 1 does not include fields such as Supplier or Planned delivery date, which correspond to later communicative events.

If the organisation has *previous business process specifications* or quality procedures, then this documentation can also be used as input for the analysis.

Using these sources and techniques, analysts identify communicative events and they specify, by means of message structures, the new meaningful information communicated in each event. This information is mainly provided by the primary

actor; in case the information of a field is provided by the primary actor, then it is attached an input *acquisition operation* (e.g. the quantity of items of a certain product that the client requests: Quantity i). Some informational elements may not be provided by the primary actor, but generated by the IS itself; in that case the acquisition operation is generation (e.g. order numbers, as well as invoice numbers, are usually pre-printed in form booklets: Order number g). With regards to the *domain*, in analysis time it is advised to use five basic types for data fields (text, number, money, date, time), and types of business objects for reference fields. It is also convenient to provide *examples* and *descriptions*.

In analysis time, it is discouraged to specify derived fields (data fields whose acquisition operation is d), as well as any other field property that is an aspect of design (see Table 2). Derived fields do not convey new information and they contribute to an unnecessary over-specification. It is convenient to postpone specifying derivation until the design stage. In case analysts decide to include design aspects in analysis time, then they should be well aware and justify their decision.

See [5] for more detailed guidelines.

4.2 Usage of Message Structures in design time

In design time, Message Structures allow establishing the traceability between analysis documentation, the IS memory model (e.g. by means of a class diagram or a relational database schema), and the specification of the user interface. Moreover, it is possible to define techniques for deriving the IS memory from requirements models, as well as techniques for systematically reasoning the interface design.

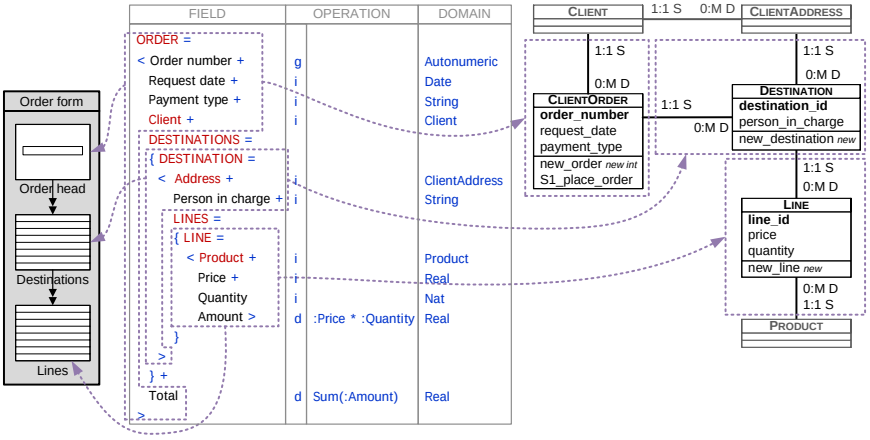


Fig. 2. Derivation of an interface design and a class diagram view that correspond to a communicative event in which an order is placed

The summarised procedure for the derivation of the IS memory is as follows[4]. First the communicative events are sorted according to their temporal precedence. Then the message structure of each event is processed in order to obtain a class diagram view. This way, the complete class diagram is iteratively created by integrating the class diagram views that correspond to all the communicative events. Fig. 2 (right-hand side) shows the derivation of the class diagram view that corresponds to a communicative event in which a client places an order. A more detailed and bigger example is available online [6].

The summarised procedure to reason the user interface is as follows. First the interface style manual is defined. Then the editing environments are identified (i.e. sets of forms or interface screens that support a set of editorially-compatible communicative events. Next, the message structures are fragmented (e.g. normalising them in first normal form) and the fragments are assigned to abstract interface structures (e.g. registry, set of registries). The abstract interface structures are encapsulated in forms. Each form is specified in detail, establishing the possible interaction and the editing facilities (e.g. filters, order criteria). The behaviour of the interface can be specified by means of trigger tables. Lastly, additional listings and printouts are specified. Fig. 2 (left-hand side) shows how the IS interface is designed in terms of abstract interface patterns. Methodological guidelines are described in detail in [3].

In design time, it is usual to specify derived fields (e.g. the total amount of each order line `Amount d (:Price * :Quantity)`). Other properties that specify aspects of design are also specified in this stage (e.g. the specification of the data field `Request date` could also include a formula that defines its initialisation value: `today()`).

5 Technological support for Message Structures

Model-driven software development (MDD) has become a reality [7]. However, the community lacks a collection of well integrated CASE tools that cover all the lifecycle, from requirements engineering to automatic software code generation, and that take advantage of model transformations in all the stages. This section presents the modelling tools that are being developed to support Message Structures. These supports extend a CASE tool under development that aims for the integration of Communication Analysis in MDD environments [8]. Two alternative development environments have been chosen; namely, Xtext and Eclipse Modeling Framework.

5.1 Support to Message Structures with Xtext

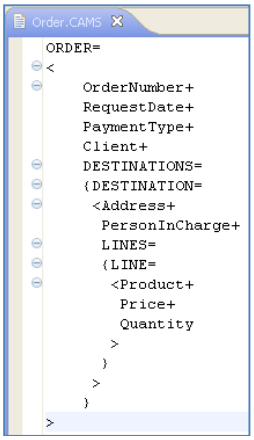
Message Structures is a modelling language based on structured text, that can be specified using the Extended Backus-Naur Form notation [5]. This characteristic facilitates the development of a domain-specific language (DSL) tool. Fig. 3.a shows

[4] We describe the derivation of class diagrams because this derivation technique is part of ongoing research. An analogous argumentation can be made for relational schemas.

the Message Structure grammar as defined in the Xtext environment, an Eclipse-based environment for the development of textual DSLs [9]. This environment allows the automatic generation of textual editors for the defined DSLs. Fig. 3.b shows the specification of the message structure ORDER, using the Xtext tool. An advantage of this environment is that it can be integrated with other Eclipse-based modelling tools.

```
grammar org.xtext.example.mydsl.CAMS with
org.eclipse.xtext.common.Terminals
generate cAMS "http://www.xtext.org/example/mydsl/CAMS"
MessageStruc:
strucName +=StrucName
(initialSubstruc +=InitialSubstruc);
StrucName:
strucName=ID '=';
InitialSubstr:
(aggregationSubstruc +=AggregationSubstruc) |
(iterationSubstruc +=IterationSubstruc);
AggregationSubstruc:
'<'(substrucList +=SubstrucList)'>';
IterationSubstruc:
'{'(substrucList +=SubstrucList)'}';
SpecialisationSubstruc:
'['(substrucList +=SubstrucList)
('|' (substrucList +=SubstrucList))']';
SubstrucList:
(substruc+=Substruc) ('+'(substruc+=Substruc))*;
Substruc:
(field +=Field) |
substrucName=ID'='(complexSubstruc+=ComplexSubstruc);
Field:
fieldName=ID;
ComplexSubstruc:
(aggregationSubstruc+=AggregationSubstruc)|
(iterationSubstruc+=IterationSubstruc)|
(specialisationSubstruc+=SpecialisationSubstruc);
```

a) DSL definition in Xtext for Message Structures



b) Example of a message structure

Fig. 3. Support to Message Structures with the Xtext environment

5.2 Support to Message Structures with Eclipse Modeling Framework

The MDD paradigm can add value to requirements models: the potential to derive from them conceptual models that can later be used for automatic code generation. With this in mind, [8] defines a process to integrate Communication Analysis in an MDD environment and it presents a metamodel that specifies part of the method. The metamodel was created using Eclipse Modeling Framework (EMF) (a class diagram was designed in UML2 Tools and its corresponding Ecore metamodel was generated).

This paper presents an extension of the Communication Analysis metamodel that allows modelling Message Structures. In Fig. 4, pre-existing metaclasses have a gray border, whereas added metaclasses have a black border. A snapshot of the tool is provided in [5].

On the one hand, the implementation in Xtext ensures the compliance with the EBNF grammar for Message Structures and it offers an editorial environment that is more efficient and usable. On the other hand, the implementation in EMF extends the CASE tool for Communication Analysis; moreover, its Ecore metamodel offers the possibility of defining model to model transformations using languages such as ATL

Transformation Language (ATL [10]) or Query/View/Transformation (QVT [11]). In any case, both implementation approaches are complementary, since both environments (Xtext and EMF) can be integrated (this is planned as future work).

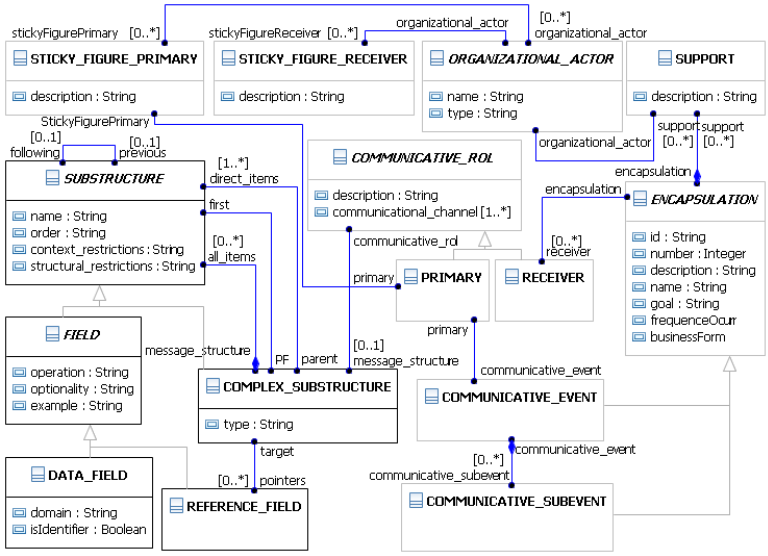


Fig. 4. Support to Message Structures with Eclipse Modeling Framework (partial view of the extended metamodel)

6 Related works

A notable ancestor of Message Structures is Backus-Naur Form (BNF). BNF is a notation for context-free grammars that was proposed during the development of the compiler Algol 60 [12]. The grammatical constructs are the sequence, represented by contiguity, and the alternative, represented by a vertical bar |. Later extensions incorporate the curly brackets { } for repetitions, and the square brackets [ ] for the alternative. Structured Analysis adapts BNF for the specification of data structures [13]. Fortuna et al. [14] extend UML Use Cases with a BNF-based notation to specify informational interfaces, with a similar intention to Message Structures.

However, no previous notation includes an operator to explicitly parenthesise sequences. This prevents the analyst from including substructures within substructures and forces the fragmentation of complex substructures. For instance, the first structure (a) presents an ambiguity that can only be avoided by fragmenting the structure in two parts or, as proposed by Communication Analysis, parenthesising the aggregation (b).

a) Vehicle=NumberPlate+Brand+Model+Motor=CubicCapacity+Valves+Fuel+Colour

b) Vehicle=NumberPlate+Brand+Model+Motor=<CubiCapacity+Valves+Fuel>+Colour

Other techniques that allow modelling the interaction between the organisational actors and the IS can be used instead of Message Structures; for instance, UML Sequence Diagrams and ConcurTaskTrees [15]. However, our experience with these techniques has shown us the convenience of using a lightweight notation, such as Message Structures, which has the advantage of allowing a fast, textual specification.

7 Discussion and future work

Information systems (IS) support organisational communication. This paper presents Message Structures, a technique that allows specifying the communicative interactions among the organisational actors and the IS. Besides detailing the grammar of Message Structures, this paper also explains its uses in analysis and design time. In analysis time, message structures facilitate the identification of communicative events (business activities that provide new significant information to the IS) and complement the business process description. In design time, message structures allow deriving the IS memory and determining the interface design.

Clear distinction between analysis and design is important in the use of Message Structures. This paper provides methodological guidelines for applying the technique to both stages. Message Structures help analysts to avoid specifying design aspects in analysis time without being conscious of them (although this can still occur if the analyst is inexperienced with the technique). The risk of work overload and over-specification of the requirements model decreases with the practice, as the nuances of the technique are internalised.

Due to the fact that Message Structures is a textual notation, it offers the advantage of allowing the specification by means of a word processor or by configuring textual fields in CASE tools. However, our experiences in industrial developments, in teaching master courses in software engineering, and in laboratory experiments (e.g. [16]), have convinced us to offer more versatile support. This paper presents two tools that support the technique. One tool is based on the Xtext technology; another one is based on the Eclipse Modeling Framework. As future work, we plan to integrate both tools. This integration will take advantage of the usability of the first technology and the capacity to support model transformation of the second technology. We are currently working on deriving conceptual models from requirements models that include message structures. This derivation is implemented using ATL Transformation Language rules.

References

1. España, S., A. González, and Ó. Pastor, Communication Analysis: a requirements engineering method for information systems, in 21st International Conference on Advanced Information Systems (CAiSE'09). 2009, Springer LNCS 5565: Amsterdam, The Netherlands. p. 530-545.

2. González, A., Algunas consideraciones sobre el uso de la abstracción en el análisis de los sistemas de información de gestión (PhD thesis) Some considerations on the use of abstraction in management information systems analysis (in Spanish), in Departamento de Sistemas Informáticos y Computación. 2004, Universidad Politécnica de Valencia, Spain.
3. España, S., Guía metodológica para especificación de interfaces gráficas de usuario basada en estructura de adquisición de datos, Methodological guide for the specification of graphical user interfaces based in communication structures (in Spanish), in Facultad de Informática de Valencia. 2005, Universidad Politécnica de Valencia, Spain. p. 240.
4. González, A., S. España, and Ó. Pastor, Unity criteria for Business Process Modelling: A theoretical argumentation for a Software Engineering recurrent problem., in Third International Conference on Research Challenges in Information Science (RCIS 2009). 2009, IEEE: Fes, Morocco. p. 173-182.
5. España, S., A. González, Ó. Pastor, and M. Ruiz, A practical guide to Message Structures: a modelling technique for information systems analysis and design.2011, Technical report ProS-TR-2011-02, ProS Research Centre, Universidad Politécnica de Valencia, Spain. http://arxiv.org/abs/1101.5341v2
6. España, S., A. González, Ó. Pastor, and M. Ruiz, Integration of Communication Analysis and the OO-Method: Manual derivation of the conceptual model. The SuperStationery Co. lab demo.2011, Technical report ProS-TR-2011-01, ProS Research Centre, Universidad Politécnica de Valencia, Spain, http://arxiv.org/abs/1101.0105v1
7. Pastor, O. and J.C. Molina, Model-Driven Architecture in practice: a software production environment based on conceptual modeling. 2007, New York: Springer. 302.
8. Ruiz, M., S. España, A. Gonzalez, and O. Pastor, Análisis de Comunicaciones como un enfoque de requisitos para el desarrollo dirigido por modelos (in Spanish), in VII Taller sobre Desarrollo de Software Dirigido por Modelos (DSDM 2010), O. Avila-García et al. (Eds.). 2010, Valencia, España. p. 70-77. http://www.sistedes.es/TJISBD/Vol-4/No-2/articles/DSDM-articulo-10.pdf
9. Behrens, H., M. Clay, S. Efftinge, M. Eysholdt, P. Friese, J. Köhnlein, K. Wannheden, S. Zarnekow, and and contributors. Xtext user guide (v 1.0.1). 2010 [cited 2010 2010-11]; Available: http://www.eclipse.org/Xtext/documentation/1_0_1/xtext.pdf.
10. Jouault, F. and I. Kurtev, Transforming models with ATL, in Satellite Events at the MoDELS 2005 Conference. 2005: Montego Bay, Jamaica. p. 128-138.
11. OMG. Meta Object Facility (MOF) 2.0 Query/View/Transformation specification (version 1.0). 2008 [cited 2010 05/2010]; Available: http://www.omg.org/spec/QVT/1.0/.
12. Backus, J.W., F.L. Bauer, J. Green, C. Katz, J. McCarthy, A.J. Perlis, H. Rutishauser, K. Samelson, B. Vauquois, J.H. Wegstein, A.v. Wijngaarden, and M. Woodger, Revised report on the algorithm language ALGOL 60. Commun. ACM, 1963. 6(1): p. 1-17.
13. DeMarco, T., Structured analysis and system specification. 1979: Yourdon Press.
14. Fortuna, M.H. and M.R.S. Borges, Modelagem informacional de requisitos, in VIII Workshop em Engenharia de Requisitos (WER 2005), J. Araújo, A.D. Toro, and J.F.e. Cunha, Editors. 2005: Porto, Portugal. p. 269-280.
15. Panach, J., S. España, I. Pederiva, and O. Pastor, Capturing interaction requirements in a model transformation technology based on MDA. Journal of Universal Computer Science (JUCS). Special issue "Designing the Human-Computer Interaction: Trends and Challenges", 2008. 14(9): p. 1480-1495.
16. España, S., N. Condori-Fernández, A. González, and O. Pastor, An empirical comparative evaluation of requirements engineering methods. Journal of the Brazilian Computer Society, 2010. 16(1): p. 3-19.