Concepts of object paradigm for an approach to modular specification of communication protocols:*

© Jörg Fischer
Otto-von-Guericke-Universität Magdeburg
Fakultät für Informatik
Institut für Technische und Betriebliche Informationssysteme.
jfischer@iti.cs.uni-magdeburg.de

June 10, 1999

Abstract

This paper deals with modularization concepts for an approach to specification of communication protocols. These concepts are related to the object oriented ones of ([EDS93]).
The syntactical representation follows the means which are in use by engineers like wiring charts.
The means which expresses communication rests upon the modularity concept which are initiated by techniques of algebraic specifications ([BG77], [BG80]) and wherein each communication partner has a common part in its specification.
A modular specification of a serial interface serves as example.

Contents

1 Introduction. **2**
1.1 Communication. 2
1.2 Modularity. 2

2 Presuppositions. **4**
2.1 Presuppositions in general. 4
2.2 Templates in this approach. 4

*Supported by the DFG under project no. NE 592/4 - 1 as well as the FireWork project.

3 On modularity. **5**
3.1 Modular specifications, qualifications and name clash. 5
3.2 Pictorial representation of templates. 5
3.3 Concepts of the object paradigm. 5
3.3.1 Generalization/Specialization. 5
3.3.2 Interfacing. 6
3.3.3 Communication. 6
3.3.4 Renamings. 6
3.3.5 Association. 7
3.4 Example: Specification of a serial interface. 7

4 Conclusion. **8**
4.1 Related work — Communication as categorical *push-out* construction. 8
4.2 Outlook. 8
4.3 Acknowledgments. 8

List of Figures

1 Communication via serial interface. 3
2 Serial interface — composed in a modular way. 3
3 Character transmission — 5 bit, 1 stop bit. 7
4 Communication as categorical *push-out* construction. 8

1 Introduction.

1.1 Communication.

The first, very typical, element of communication protocols is of course communication (see Figure 1). Communication is none but a special kind of synchronization of concurrent processes. In this approach communication partners are specified such that they have a common part which plays the role of the *interface*, intuitively. If one wishes that these partners are connected then one has to identify this common part.
There is a serious difference between the description of communication and that of MSC ([IT96]). Firstly, in MSC specifications data is followed along channels. But in our approach the timing of communication is taken into account. Secondly, in MSC a standard communication is assumed such with a FIFO buffering mechanism.

1.2 Modularity.

Practically used devices for communication have many functions. Imagine, data has to be transmitted from computer to printer. Because in most cases computers are much faster than printers one has to build a flow control which controls

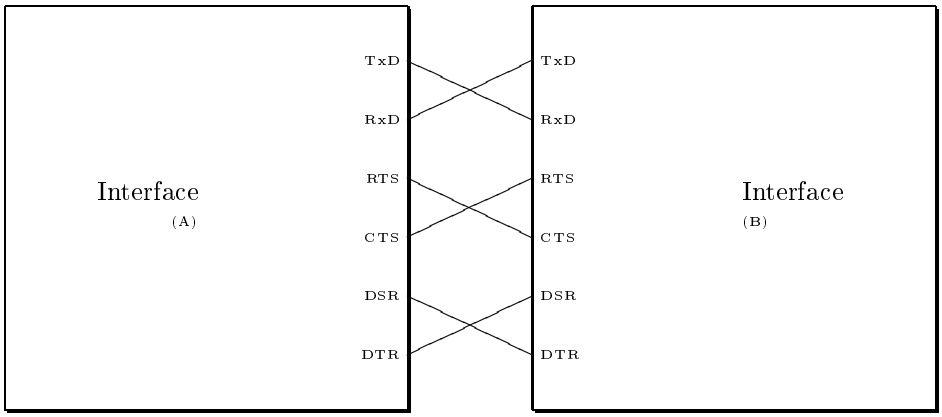


Figure 1: Communication via serial interface.

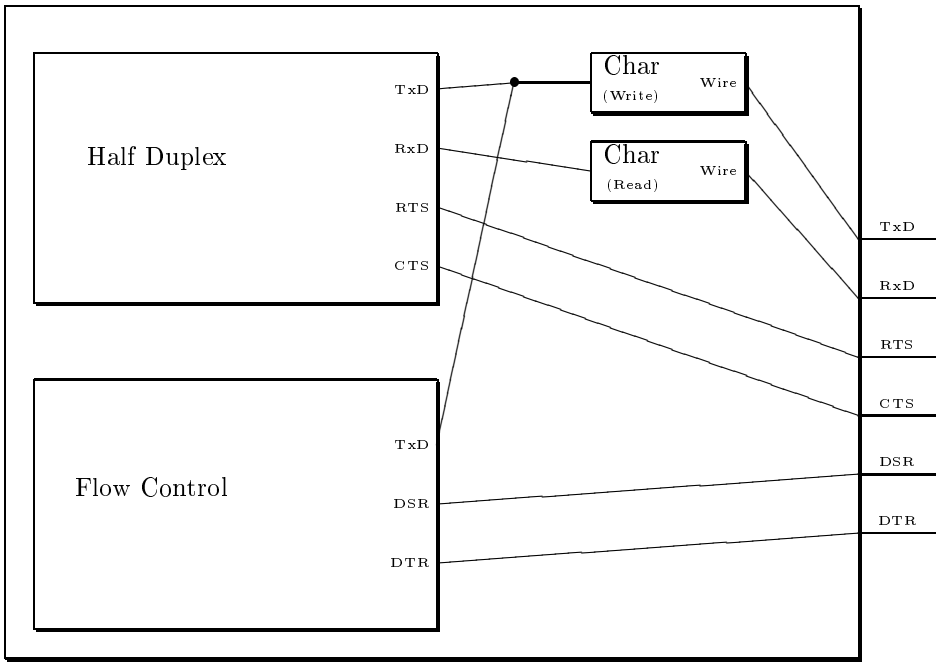


Figure 2: Serial interface — composed in a modular way.

the data flow such that the printer will not be buried under data. Another function could be semi-duplex control, see Figure 2.
Modularity in specification

- helps the reader to keep track over complex systems
- supports team work

2 Presuppositions.

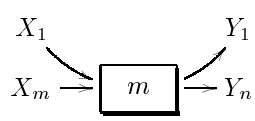
2.1 Presuppositions in general.

In this paper we suppose that there are *templates* T which have *elements* $x \in T$. Typically, that are axioms, methods and other things. If S and T are templates and all elements $x \in S$ are also element $x \in T$ then S is called a *subtemplate* of T which is denoted as $S \subseteq T$.

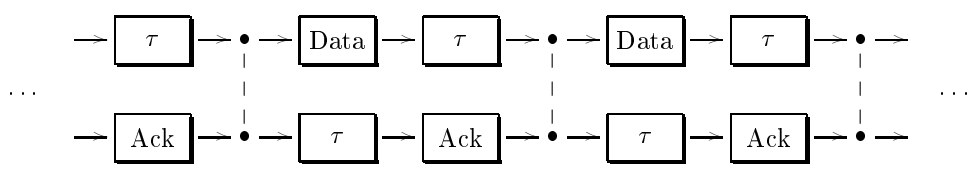
2.2 Templates in this approach.

We do not need to know the inside structure of templates. But it is useful to give to reader some details which supports the understanding of the following explanations.
At least, a template T consists in:

- a set P_T of *process types*.
- a set family $M_T = (M_{T,A,B} \mid A, B \in P^*)$ of *methods*. The methods are depicted as follows:



These methods can be composed to networks. Below is an extract of one:



Note that networks can also be infinite. Intuitively, this picture presents the course of a process by putting together the methods from left to right. There are two sequences in that picture. This expresses parallelism. The dashed lines represent simultaneity.

3 On modularity.

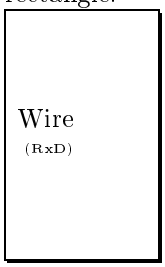
3.1 Modular specifications, qualifications and name clash.

Definition 1 (Modular specification.) *A* modular specification *is a set* Π *of templates wherein if there are two templates* $S, T \in \Pi$ *in which there is an element* $x \in S$ *and* $x \in T$ *then there is a template* $R \in \Pi$ *with* $x \in R$, $R \subseteq S$ *and* $R \subseteq T$. [1]

In order to overcome the name clash problem, that means to ensure the condition of the latter definition, and to make all templates of a modular specification independent of choose of names for its elements we introduce *qualification* of all elements of a modular specification: For each template $T \in \Pi$ of a modular specification we choose a name n_T which we call *qualifier*. Now, all elements $x \in T$ which are not in a subtemplate $T' \subseteq T$, have to be renamed to $x.n_T$.

3.2 Pictorial representation of templates.

Module structure is represented by pictures. The picture of a template is an rectangle:

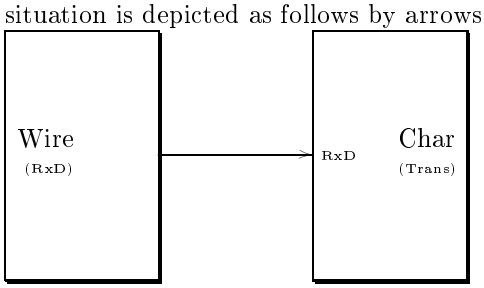


Wire is that template, RxD is its qualifier, herein.

3.3 Concepts of the object paradigm.

3.3.1 Generalization/Specialization.

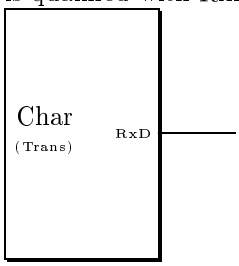
A template S is more general than a template T or a template T is more special than a template S if S is subtemplate of T. Suppose Char is a template with qualifier Trans and Wire is its subtemplate with qualifier RxD. Then this situation is depicted as follows by arrows:



[1] The latter condition avoids name clashes between elements of templates in Π.

3.3.2 Interfacing.

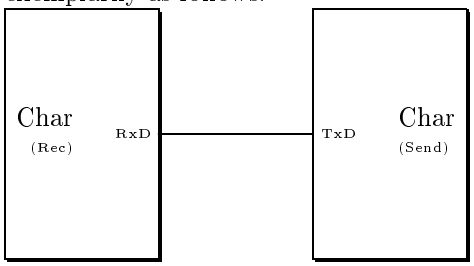
The first important concept of the object paradigm is *interfacing*. An *interface* of a template is just a subtemplate. Interfaces are that part of a template which describes the data interchange function. This situation is depicted as follows: Suppose Char is a template with a qualifier Trans having a subtemplate which is qualified with RxD:



From precise standpoint, interfacing is a generalization. Also, there could be more than one interfaces for one template.

3.3.3 Communication.

Interfacing is the base for communication. Two templates S and T do communicate if there is a common subtemplate I of them: $I \subseteq S$ and $I \subseteq T$. This subtemplate is the common interface. Pictorially, this situation is represented exemplarily as follows:



'*Aggregations*' which is taken into account in [EDS93] we regard as communication.

3.3.4 Renamings.

In the last picture a template was connected which is qualified by RxD with a template which is qualified by TxD. This is a special case of *renaming*. Templates have *inside qualifiers* as well as *outside qualifiers*. In the latter picture outside qualifier is unknown. The inside qualifiers are in the rectangles of their templates. They are known only inside these templates.
One can define outside qualifier explicitly:

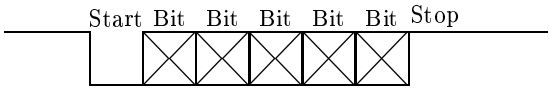
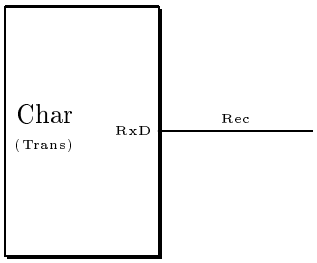


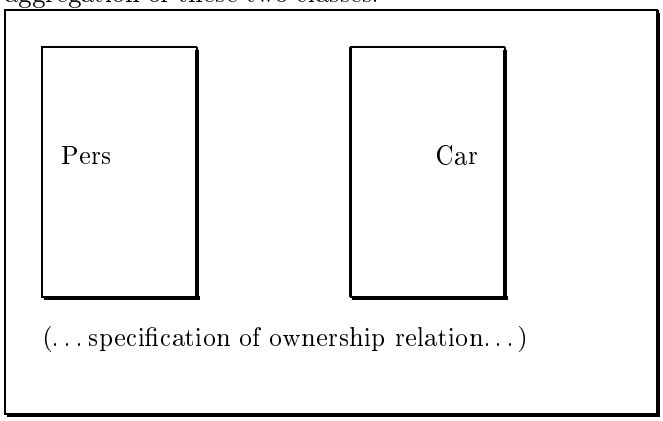
Figure 3: Character transmission — 5 bit, 1 stop bit.



Herein you have to take care of name clashes.

3.3.5 Association.

Association is the means for expressing of relations between classes, for instance the ownership relation between a class of cars and a class of persons. In this approach we describe that fact by a template which is an extension of the aggregation of these two classes:



3.4 Example: Specification of a serial interface.

Figure 1 and Figure 2 show the usage of concepts of object paradigm for presentation of communication and modular structure. In the latter picture the template Char is used to specify the serial character transmission protocol shown in Figure 3.

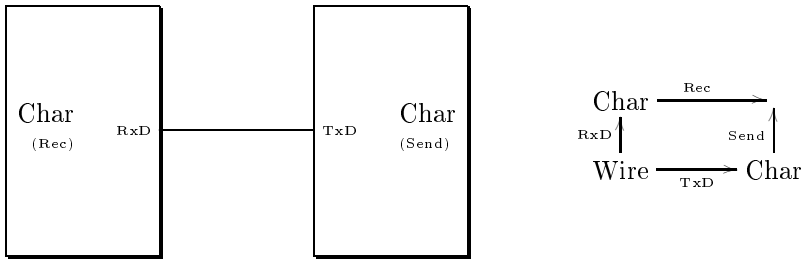


Figure 4: Communication as categorical *push-out* construction.

4 Conclusion.

4.1 Related work — Communication as categorical *push-out* construction.

As pointed out in the abstract this approach was initiated by the study of [BG77], [BG80]. The category which is in account in these papers is the subtemplate order $\subseteq$. Parametrized specifications relates to our concept '*interfacing*'. The categorical *push-out* construction, i. e. the mix of two or more specification where is a common part which is the foundation of fitting in of a specification into a parametrized specification relates to the concept '*communication*' in our framework, cf. Figure 4.

4.2 Outlook.

Specifications are the base of *verification*. That means the checking of properties whether they are fulfilled in an implementation or not. Another aim is to gather consequences by logical deduction. It would be fine when structured specifications could verified modularly. That means that properties which are proven locally are suitable for reasoning on global level.
An important requisite for that requirement we had provided in this paper. Now, we know which proposition hold in each part because we have related structuring means to operations on templates. If one had a precise formal semantics then one could answer for the question how to verify modularly.

4.3 Acknowledgments.

I am grateful to Stefan Conrad for the fruitful discussion on this paper.

References

[BG77] P. H. Burstall and J. A. Goguen. Putting theories together to make specifications. In *Fifth International Joint Conference on Artifical*

Intelligence, MIT, Cambridge, Mass., volume 2, pages 1045–1058. IJCAI-77, Department of Computer Science, Carnegie Mellon, Pittsburgh, August 1977.

[BG80] P. H. Burstall and J. A. Goguen. The semantics of clear, a specification language. In *Abstract Software Specification, Copenhagen Winter School*, volume 86 of *Lecture Notes in Computer Science*, pages 292–332. Springer-Verlag, New York, N. Y., 1980.

[EDS93] H.-D. Ehrich, G. Denker, and A. Sernadas. Constructing systems as object communities. In M. C. Gaudel and J. P. Jouannaud, editors, *TAPSOFT '93: Theory and Practice of Software Development. 4th International Joint Conference CAAP/FASE France, April 1993. Proceedings.*, number 668 in Lecture notes in Computer Science, pages 453–467. Springer-Verlag, New York, N. Y., 1993.

[IT96] ITI-TS. ITU-TS Recomendation Z.120: Message Sequence Chart 1996 (MSC96). Technical report, ITU-TS, Geneva, 1996.