An MSC Based Representation of *DiCons*

J.C.M. Baeten, H.M.A. van Beek, and S. Mauw

Department of Mathematics and Computing Science,
Eindhoven University of Technology,
P.O. Box 513, NL–5600 MB Eindhoven, The Netherlands.
{josb,harm,sjouke}@win.tue.nl

Abstract. We present a graphical MSC-based representation of the language *DiCons*, which is a formal language for the description of Internet applications.

1 Introduction

Building internet applications is not an easy task. Given the many problems involved it makes sense to investigate the use of formal methods since we think that formal methods can help to develop Internet applications more efficiently, and can help to improve the quality of applications.

Currently, a mix of different languages, at different levels, with a low degree of formality is used, e.g. Perl, C++ and Java. Recently, we have started a new line of research in order to remedy this. This has resulted in the first version of the language *DiCons* in [3] of which an extended abstract appeared as [2].

The most important feature of *DiCons* is that it is geared towards the highest level of abstraction, the communication level, and that aspects of lower levels are treated in separate parts of the language. The purpose of this paper is to give a graphical presentation of *DiCons* specifications.

The language *DiCons* focuses on a specific class of internet applications, a class we call Distributed Consensus (this explains the name of the language). This is the class of applications where several users strive to reach a common goal without having to meet face to face, nor will there be any synchronized communications between users. A central system, viz. an Internet application, must be used to collect and distribute all relevant information. Example applications are making an appointment, evaluating a paper, and selecting a winner.

Currently, we are working on the formal semantics of the language, which serves as the starting point for this paper. The papers [3,2] show the usefulness of the language in a number of examples which were first developed as MSC scenarios, and afterwards programmed in *DiCons*. MSC is the language Message Sequence Chart, that is used a lot in the telecommunications industry, as standardized in [11]. MSC has in common with *DiCons* that it is mainly concerned with the interaction between system components (here: a server and several clients) and that internal processing of information is less important.

A closer look reveals that drawing the MSCs does not leave out much information. Mostly, they contain just one scenario (or a couple of related scenarios).

R. Reed and J. Reed (Eds.): SDL 2001, LNCS 2078, pp. 328–347, 2001.

On the other hand, the examples in *DiCons* suggest that there is a main trace that admits variations occasionally. With the recent extensions of MSC [11] it could be possible to give a complete, or almost complete MSC specification of *DiCons* programs. This is the hypothesis that we investigate in this article.

There are several reasons why an MSC-like representation can have added value for a textual specification language like *DiCons*. Most important is that a visual interface can aid communication with a customer, who wants an application to be built. It is easier to understand by those not used to programming languages. Focusing on example scenarios can be very important in the initial design phase of a new application.

On the other hand, there are reasons why just using MSC as a trace description language is not enough. Most important, just describing traces can leave ambiguities, obscurities and misunderstanding about the working of an application.

In general, it is not advisable to give complete system specifications in MSC. However, it is interesting to note that *DiCons* is intended for restricted class of applications, and this makes it possible to define a complete MSC-like representation. Thus, in *DiCons* we only consider the behaviour of the server, depending on possible external stimuli and the internal state. This means we only have to define the complete behaviour of a single MSC instance. This appears to be a lot simpler than specifying the complete behaviour of several instances.

In this paper, we investigate giving an MSC-like representation of *DiCons* with comparable expressivity. A first try is to see whether MSC-2000 is powerful enough by itself, but it soon turns out that more is needed. For instance, *DiCons* involves several communication primitives that have to be represented in different ways. We see that *DiCons* has compound communications that go beyond the simple scheme of an asynchronous MSC communication. This requires extensions of MSC-2000 in order to raise the representation to the same level of abstraction. Our extensions are in the style of MSC-2000, for instance regarding the use of in-line expressions.

We are not in the business of proposing extensions of the language MSC. Rather, we look upon this work as a special application of MSC. In our experience, every (new) application domain of MSC will lead to a comparable adaptation of the language. This is due to the nature of the language MSC. On the one hand, MSC is so universal as to be applicable whenever there is a form of distribution and communication. On the other hand, the drive to express issues in the appropriate way and at the appropriate level will necessitate new features that have not been standardized by the ITU (yet). The present offering of MSC-2000 seems to have enough features already. There is a good basis of possibilities to express issues like modularization, data, time, and many more things, and it is not obvious that specific applications should lead to even more extensions of the language.

Rather, we see our work as defining a graphical layer on top of *DiCons* based on MSC, and not as an extension of MSC. This paper is exploratory: we do not give a formal graphical syntax and do not give a translation to the semantics.