A Formal Definition of Structured Analysis with Programmable Graph Grammars*

Luciano Baresi and Mauro Pezzè

Dipartimento di Elettronica e Informazione-Politecnico di Milano
Piazza Leonardo da Vinci, 32, I-20133 Milano, Italy
tel.: +39-02-2399-3638 - baresi@elet.polimi.it

Abstract. Structured Analysis has been one of the most widely used specification notations of the last decades. Friendliness and flexibility promoted its use, but informality hampered its precision and efficacy. The many proposals that tried to overcome the problem improved precision, but constrained flexibility. They propose formal and specific interpretations of Structured Analysis that, even if meritorious, do not impact on day-to-day practice. To meet the goal, formalization attempts should not try to impose particular interpretations, but they should allow users to tailor the interpretation to their current needs.

In this paper, we present a solution that merges precision and flexibility to provide a customizable and formal definition of Structured Analysis. Formalization consists of a set of customization rules and a consistency framework. Customization rules, based on graph grammars, formalize the different behaviors of notation elements by defining a mapping onto a formal model. The consistency framework groups complementary rules, which give different semantics to the same elements, and constrain the scope of each rule, that is, identifies the set of rules that may be affected by a change.

1 Introduction

Structured Analysis (hereafter, SA) is one of the notations that industry widely used in the last decades. Friendliness, accumulated experience, and flexibility promoted its use, but intrinsic imprecision hampered its efficacy. Many proposals tried to overcome the problem by complementing SA with formal methods [9]. They associate SA with formal semantics by proposing mappings from informal specifications onto formal models. For example, Semmens and Allen ([23] and Cohen ([25]) formalize SA through Z and Larch, respectively. France ([11]) proposes a translation technique from SA to SMoLC communicating processes to automatically generate formal representations of SA models. Elmstrom et al. ([17]) present a fixed set of rules to give SA a special-purpose interpretation by means of high-level Petri nets. Fencott et al. ([10]) propose semantic functions that use Z to both translate and annotate SA elements. Pethersohn et al. ([20])

* This work has been partially supported by the European Community under the ESPRIT IDERS (EIP/393) and the KIT FOR MSPEC Projects (KIT-125).
formalize SA using synchronous models. These proposals differ for the considered notation (graphical shapes), chosen formal method, and selected behavior, but they all cut flexibility by fixing particular interpretations. In contrast, informal notations, such as Structured Analysis, have always been used because of their flexibility and the many different interpretations that they permit ([7]). Thus, a valuable formalization should be as precise as a formal method and as flexible as an informal notation. Formalization efforts should consider all possible interpretations of an informal notation as well as supply means to define new feasible interpretations. Formalizing informal notations should mean formalizing neither single notations nor sets of independent notations, but notation families. A notation family comprises several related notations that share the same graphical symbols, but associate them with different semantics. New formalizations can be added to the family either by associating new interpretations with notation elements or by recombining existing ones.

A first step towards formalizing notation families has been proposed in [4], which illustrates a formalization engine that can be customized with different sets of rules to address different notations. The recent attempts of formalizing complex notation families, such as Structured Analysis, revealed the advantages as well as the limitations of the approach. We verified that the approach well supports the formal definition of complex notations, but we also identified unexpected difficulties in reusing (sharing) rules among formalized notations: The preliminary experiments resulted in different sets of rules for each notation of the family. In this paper, we propose a better solution to the problem of formalizing Structured Analysis as a notation family. It specializes the proposal in [4] by framing rules in a hierarchy that identifies complementary rules and their scope. Complementary rules formalize different interpretations for the same notation element. The hierarchy identifies the rules that could be affected by a modification.

The remainder of this paper is organized as follows. Section 2 frames the problem by summarizing Structured Analysis. Section 3 describes the approach by introducing both customization rules (Section 3.1) and the consistency framework (Section 3.2). Section 4 exemplifies the approach by presenting some alternative interpretations for making a process consume values from its input flows. Section 5 indicates the main results and ongoing work.

2 Structured Analysis

Structured Analysis ([18, 12, 24, 15, 8]) is a notation family that has evolved from mid seventies to today. In this paper, we focus on De Marco-like notations; real-time extensions are studied in [2].

Structured Analysis comprises processes, data flows, data stores, terminators, and splitting and merging points. Processes model functional data transformations. They are given different interpretations in terms of the number of consumed inputs (all/some), the number of produced outputs (all/some), and the duration of their executions. Data flows model the flow of data among processes