Abstract. We describe the concept of AutoFocus, a tool for the specification of distributed systems. AutoFocus is based on the formal development method Focus and uses graphical description formalisms embedded into its semantical framework, thus offering well-accepted notations while retaining the ability for exact consistency checks of a system under development. The tool uses a client/server architecture, with a central repository and distributed client applications in a computer network. The paper at hand focuses on the architectural and implementation-related issues of AutoFocus.

1 Introduction

In this paper, we describe AutoFocus, a tool intended to be used for specifying distributed systems on a formal basis. This basis is provided by the semantical framework of the Focus method [BDD+93], [BFG+94]. With Focus being a mathematically founded method for specifying distributed systems, AutoFocus adds graphical description formalisms to this framework that are well-known and well-accepted in industrial systems development, thus offering advantages of both approaches.

The following section 2 briefly outlines the basis of AutoFocus, the graphical description formalisms, whereas section 3, the main part of this paper, focuses on the implementation-related aspects of the tool.

2 Description Techniques

2.1 System Structure Diagrams – SSDs

System structure diagrams describe the static aspects of a distributed system by a network of interconnected components, exchanging data over channels. Each component has a unique identifier and a set of input and output channels.
attached to it. Each channel is characterized by a channel identifier and a data
type describing the set of messages that may be sent on it. Thus system structure
diagrams provide both the topological view of a distributed system and the
signature (syntactic interface) of each individual component.

Graphically, system structure diagrams are represented as graphs, where rect-
angular vertices symbolize components and arrow-shaped edges stand for chan-
nels.

2.2 Datatype Definitions

The types of the data processed by a distributed system are defined in a textual
notation. We use the basic types and data type constructors from the functional
programming language Gofer [Jon93] for this purpose. Data defined in this way
may be referenced e.g. in SSDs and STDs.

2.3 State Transition Diagrams – STDs

State transition diagrams, which are extended finite automata similar to the
concepts introduced in [GKRB], are used to describe the dynamic aspects, i.e.
the behaviour, of a distributed system and of its components. Each system com-
ponent can be associated with an automaton. Each transition has a set of anno-
tations: a pre- and a postcondition, encoded as predicates over the data state of
the system, which are satisfied before and after the transition, and a set of input
and output patterns describing the messages that are read from or written to
the input and output channels of the corresponding component.

Graphically, automata are represented as graphs with labeled ovals as states
and arrows as transitions.

2.4 Extended Event Traces – EETs

Extended event traces are used to describe exemplary system runs from a com-
ponent-based view. We use a notation similar to ITU-standardized message se-
quence charts (MSCs) with core concepts taken from MSC'96 [Int96].

2.5 The Concept of Hierarchy

A common property shared by all of AUTOFOCUS' graphical description for-
alisms is the concept of hierarchy. Both structure diagrams and state transi-
tion diagrams – which are essentially graphs – as well as extended event traces
allow hierarchical refinement: In a structure diagram, a system component may
be viewed as a conceptual unit comprising a network of sub-components. In the
same way, a state in a state transition diagram can be refined by another STD
which details this state. In EETs, we allow so-called "boxes" as an abbreviating
notation for parts of system runs specified in other EETs.