Gesellschaft
für Informatik e.V.

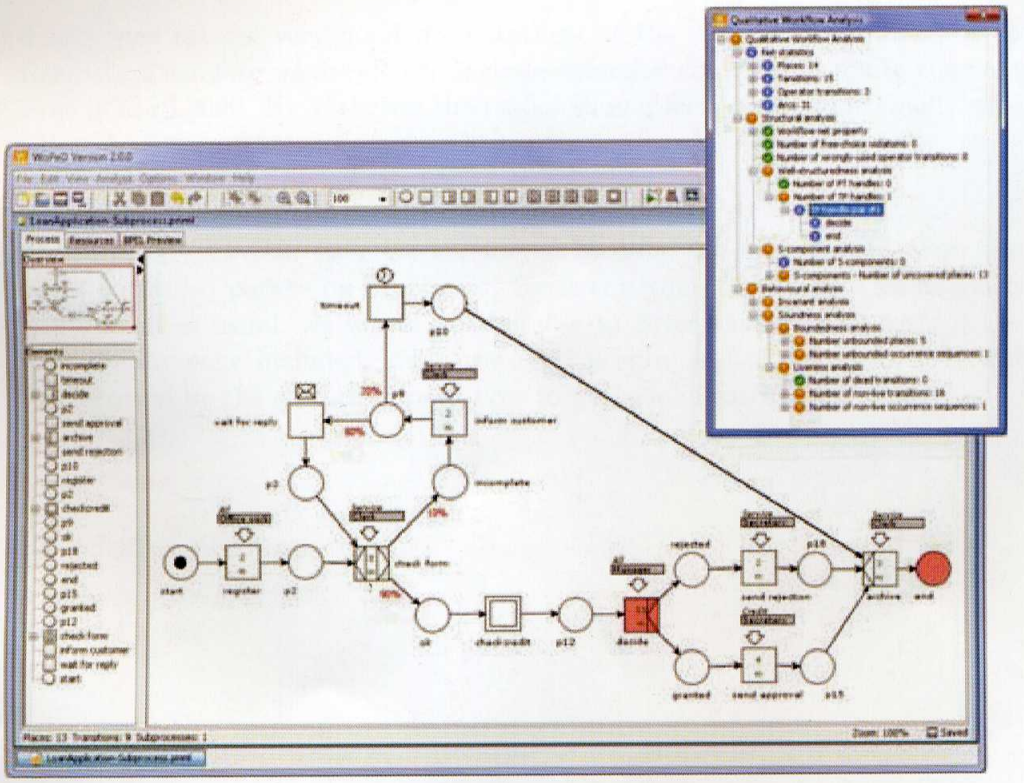
Petri Net Newsletter

Newsletter of the
Special Interest Groups on

**Petri Nets and
Related System Models**

Volume 75
ISSN 0931-1084

October 2008



Evinrude: A Tool to Automatically Transform Program's Sources into Petri Nets

Jean-Baptiste Voron & Fabrice Kordon

Université Pierre & Marie Curie
UMR CNRS 7606, LIP6 MoVe
4, place Jussieu, Paris, F-75005 France
jean-baptiste.voron@lip6.fr & fabrice.kordon@lip6.fr

Abstract. Model checking is a suitable formal technique to analyze parallel programs' execution in an industrial context because automated tools can be designed and operated with very limited knowledge of the underlying techniques. However, the specification must be given using dedicated notations that are not always familiar to engineers (so far, model checking on UML raises complex problems that will not be solved immediately).
This paper proposes an approach and its implementation as a tool to perform transformation of C source code into Petri nets, a suitable specification for model checking. To overcome the complexity of the resulting specification, we focus on specific aspects of the program. Hence, we never model the entire processed program, but only its relevant parts. In this paper, we will apply this approach on some examples using our tool: *Evinrude*.

1 Introduction

Behavioral analysis of concurrent systems cannot be completed anymore using only "traditional" test-based approaches. First, their complexity often makes impossible to cover a significant part of the state space by simulation. Second, testing concurrent systems is not trivial and may lead to complex problems like probe effects [1]. To overcome these limitations, it is now recognized for many years that formal methods are of interest since they provide more trustable and mathematically founded information [2, 3].

Among the available formal verification techniques, model checking is particularly interesting due to its potential for full automation as well as its error reporting capabilities [4, 5]. So, neither a long training nor a long practice are required from engineers, using model checkers to extract execution paths leading to undesired behaviors.

However, the problem is about the way engineers design specifications too. Most model checkers require formal specifications as inputs : Automata [6], Promela [7], Petri nets [8], etc. These input formalisms may be difficult to learn. They usually also propose a low level of abstraction not adapted to their use in the context of industrial-size projects without extensive practice.