Thesis proposal:

Rigorous analysis of combined software processes via runtime verification

by

Mark McDermott

A04474453

One the main goals of software engineering is to be able to evaluate, compare and improve software processes to increase software qualities produced by such processes. To be able to do so rigorously, in automated fashion, a process programming approach was suggested [1]. A number of process languages and associated enactment systems have appeared over the years that either tried to adapt regular programming languages to process definition or suggested new process oriented languages. One important feature of a language for rigorous analysis is rigorously defined semantics. Such rigorously defined semantics enables one to use existing program analysis algorithms to verification of processes. While there is a great deal of work in the area of process and programming we will focus on the existing process language called Little-JIL and its associated enactment system [2].

This has already been used to rigorously analyze software processes: e.g. [3], [4]. In these works a process definition was mapped to an input language of a data flow analysis system called Flavors and then analyzed against given properties.

In this work we will focus on runtime verification of a software process as it is enacted by Little-JIL enactment system. Similar systems exists for programming languages, e.g. Java Pathfinder for Java programs. Yet this has not been done for software process definitions.

In addition, the proposed process analysis approach will allow to analyze several combined processes enacted together in a single resource environment. Both in the software development domain and other domains, e.g. medical processes, it is quite often the case that several processes developed by different developers have to run in the same environment and comply with the same set of properties. For instance, in software development domain, we might have a software