A major part of research in process modeling languages and, even more, in the workflow area has focussed on what I call "data-flow-based" languages. This concerns not only data-flow-based diagrams but also more sophisticated approaches like Petri-Nets, Event-Transition Diagrams, Statecharts and the like.

Although projects like Marvel or Merlin have started early on with descriptive approaches based on a rule-like definition and enactment of process definitions, the power of those approaches has not really been fully explored, as it seems.

The papers accepted for this session of the workshop indicate that descriptive approaches still have a lot to contribute to software process technology. The paper by Andreoli, Meunier, and Pagani explores the rule-like notion in particular for expressing coordination and concurrency. The position paper by Cisse and Pimenta discusses the fuzzyness of requirement specifications and the need to formalize those specifications in such a way that they are confirmed by all stakeholders. The technology applied to solve this problem is an actor model which describes the requirements definition process as a set of autonomous cooperating actors. The long positon paper by Alloui, Latrous, and Oquendo presents a modal logic-based approach including operators like belief, goal, and time to deal especially with the problem of process model evolution, i.e. "changes-on-the-fly".
Process Enactment and Coordination

Jean-Marc Andreoli, Jean-Luc Meunier, Daniele Pagani

Rank Xerox Research Centre
6 Chemin de Maupertuis
38240 Meylan (France)
{Andreoli, Meunier, Pagani}@grenoble.rxrc.xerox.com

Abstract. This paper investigates the relationship between systems to enact software processes and systems to coordinate distributed, heterogeneous and concurrent objects. In particular, we describe in detail how one of these coordination systems—systems—the “Coordination Language Facility” (CLF), developed at the Rank Xerox Research Centre—can be used to model and execute a sample software development process: bug reporting. The main advantages of using CLF are: i) language facility that allows to dynamically change both the core process model and the application-specific process templates; ii) modular architecture that allows to easily reconfigure, migrate and replicate each process component in a distributed, heterogeneous environment.

1 Introduction

The field of Software Process Technology (SPT) pioneered the research on how computer systems can support the software development process. Several systems originally designed to support software processes have been applied to the domain of office procedures and business processes (e.g., ProcessWeaver [13]). On the other hand, research in Office Information Systems (OIS) and Computer-Supported Cooperative Work (CSCW) led to a number of research prototypes and products aiming at supporting “office work” and “knowledge work” within customer-focused business processes, usually referred to as “workflow management systems”. Some workflow systems have been applied to support software development processes, e.g. [23]. As Chroust argued [7], the requirements of software processes and business processes are indeed different; however, there is a large degree of overlap in the coordination primitives required to orchestrate people (workers, programmers, analysts, managers, etc.) and software tools (document management systems, databases, legacy applications, compilers, debuggers, etc.).

Some authors ([18], [15]) already addressed the key role of coordination in software development. On the other hand, the research on the coordination of concurrent (object-oriented) systems led to the definition of several systems to model and execute the coordination of distributed and heterogeneous software components ([9], [10]). In this paper, we investigate how one of these coordination systems—the “Coordination Language Facility” (CLF) [1], developed at the Rank Xerox Research