2
Representing Knowledge About Information Systems in Telos

John Mylopoulos, Alex Borgida, Matthias Jarke and Manolis Koubarakis

ABSTRACT Telos is a language intended to support the development of information systems. The design principles for the language are based on the premise that information system development is knowledge-intensive and that the primary responsibility of any language intended for the task is to be able to formally represent the relevant knowledge. Telos' features include: an object-centered framework which supports aggregation, generalization and classification; a novel treatment of attributes; an explicit representation of time; and facilities for specifying integrity constraints and deductive rules. The language is appropriate for representing knowledge about a variety of worlds related to an information system, such as the subject world (application domain), usage world (user models, environments), system world (software requirements, design), and development world (teams, methodologies).

2.1 Introduction

Language facilities have been a key vehicle for advances in software productivity since the introduction of assembler in the early 50's, the first high level programming languages in the mid-50's, and the languages supporting encapsulation/modularization in the 70's. But programming accounts for only a small fraction of the total effort and cost of producing a software system.

This chapter describes a language that is intended to support software engineers in the development of information systems throughout the software lifecycle.
This language is not a programming language. Following the example of a
number of other software engineering projects, our work is based on the premise
that information system development is knowledge-intensive and that the primary
responsibility of any language intended to support this task is to be able to formally
represent the relevant knowledge. Accordingly, the proposed language is founded
on concepts from knowledge representation [BL85].

How is a knowledge representation language different from other types of
languages, such as programming or design languages, formal languages or natural
languages? According to [BL85] (pages xiv–xv):

In order to have an explicit knowledge base, a system must rely on some well-specified language for encoding its beliefs. That role is played by a knowledge representation language. Beyond that, in just about all imaginable cases of interest, a system will be concerned with more than just the literal set of sentences (or frames, or production rules, or whatever) representing what it knows. A representation system must also provide access to facts implicit in the knowledge base. In other words, a representation component must perform automatic inferences for its user.

The ingredients of a knowledge representation language include a (formal) notation, and a deductive mechanism for drawing inferences from a body of statements (the knowledge base) represented in that notation. In addition, there is a need to assign some sort of “meaning” to statements – the semantics of the notation; this meaning must be respected by the deductive process. Finally, to be effective in large projects, a knowledge representation language must offer facilities to structure and organize the knowledge base.

The language presented in this chapter is called Telos. What knowledge needs to be represented about an information system? To begin with, knowledge about the environment within which the system will function and how the system is expected to interact with that environment. Second, the kind of information the system will be expected to store and the meaning of that information with respect to its intended subject matter. Third, knowledge about the design and implementation of the information system, which can be used during initial system development as well as during system maintenance. Fourth, knowledge about design decisions that led to the particular design/implementation, along with appropriate justifications that relate these decisions to performance or other non-functional requirements. Finally, information on the development process itself that led to the system,

---

3 See [RW86] for a survey of knowledge-based software engineering projects.
4 There is a tension between the increased “expressive power” of a notation – the ability to express certain facts and make certain kinds of deductions – and the complexity of the computations involved.
5 From the Greek word τέλος which means end; the object aimed at in an effort; purpose.