UC Merced
Proceedings of the Annual Meeting of the Cognitive Science Society

Title
A cognitive Approach to the Elicitation of Skills and Specifications

Permalink
https://escholarship.org/uc/item/1sp3z69c

Journal
Proceedings of the Annual Meeting of the Cognitive Science Society, 18(0)

Authors
Lenart, Mihaly
Pasztor, Ana

Publication Date
1996

Peer reviewed

eScholarship.org Powered by the California Digital Library
University of California

A Cognitive Approach to the Elicitation of Skills and Specifications

Mihaly Lenart and Ana Pasztor
University of Kassel, Dept. of Architecture
Henschelstr. 2, 34109 Kassel, Germany
School of Computer Science, Florida International University
University Park, Miami, Fl 33199
michael@architektur.uni-kassel.de; pasztora@scs.fiu.edu

Introduction

There are two major questions underlying any (human or computer supported) problem solving process: "what" is it the problem solver has to do, and "how" to accomplish this. Answering these questions requires information gathering. In the first case, we look for *specifications*, i.e. the description of constraints, features, and goals of the task at hand. In the second, for procedures or *skills* that are necessary to achieve the goals. Specifications play an important role in design and manufacturing, but also in communication. Skills are the prerequisit for executing or performing tasks. The purpose of the elicitation process is to tranfer or make explicit knowledge of "what" people have in their mind and "how" they perform tasks. Eliciting skills and specifications is fundemantal to human communication and learning. Moreover, knowledge elicitation or acquisition is a major bottleneck in (human or computer supported) problem solving.

Walk-Through

Our first aim is to develop a strategy for eliciting skills and specifications such that the resulting description will be *implementable*. In fact, implementable skills or specifications might lead to computer programs (e.g. knowledge based or expert systems) for solving problems or generating designs. Our second aim is to make the strategy itself implementable and develop tools supporting the elicitation process. Elicitation is done by walking systematically through the process with all the participants and ask questions until a complete, consistent, and implementable description emerges. Although the "walk-through" requires domain knowledge, it is a general purpose, domain-independent strategy. Currently, we are in the process of implementing this strategy i.e. developing an elicitation tool. The "walk-through" is defined recursively (see **LOOP** below). We assume here only two participants, the user and designer. The process starts with the user entering the initial description of the task or skill, say SPEC. SPEC initiates the construction of a specification tree, which becomes the primary description of the skill or specification. After this, the designer defines the operations s/he has to perform to satisfy SPEC. These will be stored in $OP_1, \ldots, OP_n$.

(begin LOOP) In general, for each operation OP to be performed by the designer there are three kinds of required information: rules, variables, and suboperations.

Rules: Here the designer lists all rules related to OP in his/her knowledge base. Each rule is of the form $R \rightarrow R'$ and is compared with the current user specification SPEC. If R is a requirement contained in SPEC, then the user is asked whether R' is consistent with the requirements in SPEC. If the answer is "no", the user changes SPEC to remove the inconsistency, and the "walk-through" backtracks as many steps as needed to accommodate the changed specification.

Variables: Here the designer lists all variables (slots) which need to be given values in order to perform OP. In order to chunk the non-implementable requirements of the current specification SPEC into implementable requirements, the user is asked whether any of the requirements relate to the variables. If the answer is "yes," the user instantiates the relevant variables. Variables which are not instantiated by the user are given default values.

The instantiation of the variables by the user is controlled by the so called Chunking Agents by taking a requirement and checking whether it is common knowledge, quantifiable, or can be calibrated. They recognize common knowledge by looking it up in a domain specific database. A requirement is quantified if it has a *standard* numeric measure assigned to it. Calibration means to match or adjust two value systems to a given scale.

Suboperations: Here the designer lists all suboperations needed to be carried out in order to perform OP. **(end LOOP)**

Now **(begin LOOP)** through **(end LOOP)** is repeated for another operation, until all operations are exhausted and all relevant variables have implementable values. The user is presented with the generated specification tree. If s/he thinks that any of its non-implementable requirements has not been fully redefined using just the variables presented during the "walk-through" process, the Dialogue Agent with the system once again helps find the missing variables and give them values as before. The specification tree is expanded accordingly.

Implementation

For the development and testing of this tool, we analyze and compare four elicitation tasks: with the Department of Biomedical Engineering at the University of Miami the design of a biomedical device for a group of handicapped people, to help them walk with less energy; with the Mechanical Engineering Department at Florida International University, the design of a heart model for testing and teaching purposes; with the Atlantic Meteorology and Oceonagraphic Laboratory/NOAA the design of a database for data collected from drifting buoys; and with the Bascom Palmer Eye Institute of the Universty of Miami the design of an automated system for experimentations on dark-adaptation.