ARIES - Acquisition of Requirements and Incremental Evolution of Specifications

Nancy A. Roberts
Rome Laboratory /C3CA
525 Brooks Road
Griffiss AFB, NY 13441-4505
EMAIL: nancyr@aivax.rl.af.mil

Abstract

This paper describes a requirements/specification environment specifically designed for large-scale software systems. This environment is called ARIES (Acquisition of Requirements and Incremental Evolution of Specifications).

ARIES provides assistance to requirements analysts for developing operational specifications of systems. This development begins with the acquisition of informal system requirements. The requirements are then formalized and gradually elaborated (transformed) into formal and complete specifications.

ARIES provides guidance to the user in validating formal requirements by translating them into natural language representations and graphical diagrams. ARIES also provides ways of analyzing the specification to ensure that it is correct, e.g., testing the specification against a running simulation of the system to be built.

Another important ARIES feature, especially when developing large systems, is the sharing and reuse of requirements knowledge. This leads to much less duplication of effort. ARIES combines all of its features in a single environment that makes the process of capturing a formal specification quicker and easier.

1.0 Introduction

In 1988, Rome Laboratory funded a joint effort between the University of Southern California Information Sciences Institute and Lockheed Sanders Inc. to develop a project named ARIES (Acquisition of Requirements and Incremental Evolution of Specifications). The goal of the ARIES effort was to build a requirements/specification environment to become part of a larger Rome Laboratory program known as the Knowledge-Based Software Assistant (KBSA). In fact, ARIES was a technical follow-on to two earlier Rome Laboratory efforts, the KBSA Requirements Assistant and the KBSA Specification Assistant.

KBSA is a knowledge-based system that addresses the entire software life cycle. It takes a revolutionary approach to solving the software problem by formalizing and automating the processes of software development as well as the products. The impact of this process formalization is that software will be derived directly from requirements and specifications through a series of meaning-preserving and evolutionary transformations. Transformations ensure that specifications that evolve are correct.

The current requirements-engineering process creates many challenges for the ARIES system to overcome. One problem, especially true in large systems, is managing and coordinating the work of requirements analysts' working alone or in groups on a

project. Once the requirements have been captured, the next question is how to bridge the gap between the initial requirement concepts and the formal and complete specifications. There must also be some validation techniques to ensure the captured specification is correct relative to requirements. Lastly, there is an immense amount of information used in large systems. This poses the question, "Can any part of this work be reused?" All of these problems are addressed in ARIES and are described below.

2.0 The ARIES Environment

ARIES is an environment to assist in requirements engineering. One of the main ideas behind ARIES and KBSA has been to automate some of the more mechanical processes of developing software while leaving the high-level decisions up to the analyst. Thus, the ARIES environment is centered around its knowledge base.

The ARIES knowledge base contains knowledge about domains, reusable requirement/specification components, and descriptions of specific systems being developed. In addition, ARIES provides several tools that interact with the knowledge base and that complete the processes involved with requirements engineering. These processes include acquisition, review, evolution, reasoning, and reuse.

2.1 Requirements Acquisition & Review

The acquisition process begins with an analyst or groups of analysts inputting information about the system to be built. This information includes knowledge about application domains, system components and design processes. The analyst may either type in textual information or input information into graphs ie. data-flow diagrams, information-flow diagrams, hierarchies, taxonomies or spreadsheets. All the information input into ARIES makes up the underlying system representation.

The review process looks at the captured requirements and makes sure that they match up with what the user wants. This process uses many of the same tools used in capturing the requirements, but in reverse. A paraphrasing tool transforms specifications into their respective English descriptions. Diagrams, informal text, and formal specifications are "views" into the underlying system representation. Information in one view may be reviewed by looking at the information in another view. The review process shows the errors made when multiple analysts are working on the system. Inconsistencies are caught early that may have caused larger problems later.

2.2 Evolution

The ARIES environment helps to bridge the gap between requirements and specifications. When building ARIES, the developers took into account the fact that most people don't like to work with formal specifications. ARIES uses unique tools called evolutionary transformations [that help convert requirements into specifications. Transformations are the formal operations of the software engineering process. Evolution transformations, when invoked, modify some aspects of the system while leaving other parts unchanged. The transformations check the specification to ensure that the change is consistent with other decisions and automatically propagates changes throughout the entire specification. ARIES generates specifications in Reusable Gist that can be used for reasoning and execution of simulations.

2.3 Reasoning

Once a specification is captured in ARIES, the next step is to use reasoning techniques to test its validity. ARIES provides for two different types of reasoning: static analysis and dynamic analysis [1].

Static analysis tools look at the current state of the system to find inconsistencies. These tools are provided to analysts in the form of functions that can be invoked on folders or folder components. The most commonly used static analysis tool is called the type checker. It reports errors such as unresolved references, duplicate definitions and type mismatches in expressions.

Dynamic analysis tools help the analyst to get an idea of the system's behavior. The main tool for dynamic analysis is simulation. The simulation facility translates descriptions of the system into an executable simulation. Its purpose is to uncover undesirable behaviors and to ensure the presence of desirable behaviors in the current specification. Simulations are usually run on only a part of the problem at a time to make it easier to see where the inconsistencies come from.

2.4 Structure and Reuse

ARIES structures its knowledge base is into objects called "workspaces" and "folders". A workspace is a separate area where analysts can work on their part of the system. An analyst can have more than one workspace to work on two solutions to a problem or to work on a different domain. Each workspace consists of a set of folders. Folders contain formal and informal information about the system. By organizing the knowledge base into workspaces and folders, information can be shared or kept separate.

ARIES' reuse deals with the reuse of requirements knowledge or domain concepts and is based on reusing information in folders. This means creating specialization hierarchies of certain information. The user has the power to control how much or how little the folders are reused.

ARIES features two different types of reuse: reuse in the same domain and reuse across different domains. A simple example of reuse in the same domain is the inheritance of common terminology. An example of reuse across different domains is the reuse of information between an air-traffic control domain and road traffic control domain. Both types of reuse help to control work duplication and the overall size of the software being developed.

3.0 Conclusion

The ARIES environment makes it easy to enter and modify requirements and specifications through the use of natural language paraphrasing, graphical diagrams and formal specifications. ARIES also provides for the constant checking of the requirements and specifications for consistency. An analyst can review requirements information by switching views to get different perspectives. Reasoning about specifications can be done through the use of tools such as a type checker and executable simulations. The organization of the ARIES knowledge base allows for information and work to be reused, and all the tools combined will aid an analyst in the complicated process of requirements engineering. Thus, the ARIES environment integrated in the Knowledge-Based Software

Assistant promises to make great improvements in solving the software problems of today and the future.

References

1. Harris, David R., W. Lewis Johnson, Kevin M. Benner, Martin S. Feather, "ARIES: The Requirements/Specification Facet for KBSA" Final Technical Report

2. Johnson, W. Lewis, Martin S. Feather and David R. Harris, "The KBSA Requirements/Specification Facet: ARIES", Proceedings of the 1991 Knowledge-Based Software Engineering Conference, pp 53-64.