View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by NASA Technical Reports Server

Analysis, Simulation, and Verification of Knowledge-Based, Rule-Based, and Expert Systems

This method allows valid updates to be made quickly, efficiently, and without corruption of the existing rule base.

Goddard Space Flight Center, Greenbelt, Maryland

Mathematically sound techniques are used to view a knowledge-based system (KBS) as a set of processes executing in parallel and being enabled in response to specific rules being fired. The set of processes can be manipulated, examined, analyzed, and used in a simulation. The tool that embodies this technology may warn developers of errors in their rules, but may also highlight rules (or sets of rules) in the system that are underspecified (or overspecified) and need to be corrected for the KBS to operate as intended.

The rules embodied in a KBS specify the allowed situations, events, and / or results of the system they describe. In that sense, they provide a very abstract specification of a system. The system is implemented through the combination of the system specification together with an appropriate inference engine, independent of the algorithm used in that inference engine. Viewing the rule base as a major component of the specification, and choosing an appropriate specification notation to represent it, reveals how additional power can be derived from an approach to the knowledge-base system that involves analysis, simulation, and verification.

However, in a complex rule base that may have taken years, if not decades, to build, expecting users to have in-depth understanding of the rules that make up the system is not practical. This innovative approach requires no special knowledge of the rules, and allows a general approach where standardized analysis, verification, simulation, and model checking techniques can be applied to the KBS.

The rules of the system are likely written in a particular syntax, the possibilities for which include a language or grammar used by a particular inference engine, logic rules (written in Prolog or another logic programming or declarative programming language), propositional or predicate calculus, Horn clauses, or some form of structured English. A translator is required to translate these into the grammar of a tool that is used (and for which there is a prototype) to convert to a formal language that is process-based: that is, that recognizes that processes (or units of computation) are key components of a system.

All systems, regardless of how trivial, involve at least two processes, one being the environment in which the system is executing. Processes being enabled are analogous to rules firing; “deadlock” is equivalent to contradictions or internal inconsistencies existing in the rule base; “livelock” is equivalent to having rules that are unspecified; “top” is equivalent to overspecification; and “bottom” is equivalent to rules being underspecified. The system having been translated to the appropriate formal language, tools that already exist for that language may be applied to the analysis of the system. For example, if CSP were used as the formal language, the laws of CSP could be used to prove the absence, or otherwise, of contradictions, to pinpoint unimplemented rules, or to transform rules into a more efficient form. Then other available tools could be used for simulation and model checking.

This work was performed by Mike Hinchey, James Rash, John Erickson, and Denis Gracanin of Goddard Space Flight Center and Chris Rouff of SAIC. Further information is contained in a TSP (see page 1). GSC-14942-1

Core and Off-Core Processes in Systems Engineering

This methodology can reduce the difficulty of coordinating multiple systems-engineering activities.

NASA’s Jet Propulsion Laboratory, Pasadena, California

An emerging methodology of organizing systems-engineering plans is based on a concept of core and off-core processes or activities. This concept has emerged as a result of recognition of a risk in the traditional representation of systems-engineering plans by a Vee model alone, according to which a large system is decomposed into levels of smaller subsystems, then integrated through levels of increasing scope until the full system is constructed. Actual systems-engineering activity is more complicated, raising the possibility that the staff will become confused in the absence of plans which explain the nature and ordering of work beyond the traditional Vee model.

Core activities are those that produce a top-down decomposition and bottoms-up integration of a system in order of increasing time. Examples of core activities are definition of requirements, design, acquisition, and integration. Because of ordering according to time, these activities are often readily understood and depicted by use of such elementary graphical aids such as timelines and Gantt charts.

Off-core activities are other systems-engineering activities that add desirable qualities to a system solution, but are not directly involved in decomposition and