A Rule-Based Method for Scalable and Traceable Evaluation of System Architectures

Daniel Selva
Post-doctoral associate
MIT Department of Aeronautics and Astronautics
77 Massachusetts Ave Room 33-409
Cambridge, MA 02139
dselva@mit.edu
T: (617)-682-6521

Bruce Cameron
Lecturer
MIT Engineering Systems Division
77 Massachusetts Ave Room 31-161b
Cambridge, MA 02139
bcameron@mit.edu
T: (617)-253-8985

Edward F. Crawley
Full Professor
MIT Department of Aeronautics and Astronautics
77 Massachusetts Ave Room 33-413
Cambridge, MA 02139
crawley@mit.edu
T: (617)-253-7510

Abstract
Despite the development of a variety of decision-aid tools for assessing the value of a conceptual design, humans continue to play a dominant role in this process. Researchers have identified two major challenges to automation, namely the subjectivity of value and the existence of multiple and conflicting customer needs. A third challenge is however arising as the amount of data (e.g., expert judgment, requirements, and engineering models) required to assess value increases. This brings two challenges. First, it becomes harder to modify existing knowledge or add new knowledge into the knowledge base. Second, it becomes harder to trace the results provided by the tool back to the design variables and model parameters. Current tools lack the scalability and traceability required to tackle these knowledge-intensive design evaluation problems. This work proposes a traceable and scalable rule-based architecture evaluation tool called VASSAR that is especially tailored to tackle knowledge-intensive problems that can be formulated as configuration design problems, which is demonstrated using the conceptual design task for a laptop. The methodology has three main steps. First, facts containing the capabilities and performance of different architectures are computed using rules containing physical and logical models. Second, capabilities are compared with requirements to assess satisfaction of each requirement. Third, requirement satisfaction is aggregated to yield a manageable number of metrics. An explanation facility keeps track of the value chain all along this process. This paper describes the methodology in detail, and discusses in particular different implementations of preference functions as logical rules. A full-scale example around the design of Earth observing satellites is presented.

Keywords Conceptual design, design evaluation, requirement traceability, rule-based systems.

1 Introduction

A common goal of architecting and conceptual design is to make the high-level design decisions that will define the main functions, the forms that will perform these functions, and the relationships between these forms and with the surrounding context. The role of architectural decisions and their influence on engineering systems and products has been extensively studied (Crawley et al., 2004; Ulrich, 1995). Part of their importance has been quantified: 1) 70-80% of the lifecycle cost is usually fixed after the conceptual design phase; 2) the cost of repairing design defects also usually increases at least by a factor of three after this phase (Haskins, 2006; Smith & Reinertsen, 1997). Although the conceptual design and system architecting communities seem to be different, they certainly use similar tools and methods for exploring the design or architectural space. The work presented in this paper, introducing the VASSAR methodology (for Value Assessment of System Architectures using Rules),applies to both processes.

The importance of conceptual design and architecting has fostered interest in developing computational tools to support it (Campbell, Cagan, & Kotovsky, 1999; Chandrasekaran, 1989; Hauser & Clausing, 1988; Koo, Simmons, & Crawley, 2009; Radovcic & Remouchamps, 2002; Shah & Rogers, 1993; Suh, 1998; Ziv-Av & Reich, 2005). Some emphasis has been recently put into the development of flexible tools that can tackle a variety of problems and can be reused from project to project, such as BOSS-Quattro (Radovcic & Remouchamps, 2002) or FIPER (Rohl et al., 2000). Our work shares this concern, but applies to a different subset of design problems, namely system architecture problems, as will be described in Section 3.1.

A simple task analysis reveals the advantages of incorporating computational tools to conceptual design: humans are creative, have common sense, and have the ability to look at a conceptual design holistically; conversely, they can only consider a limited set of designs due to computational limitations, and have judgment biases due to their expertise. Computers have much higher computational power and can ensure rigor and consistency, but they lack creativity and the ability to see the "big picture". Therefore, the optimum level of automation in conceptual design is a compromise between a completely automatic process and a completely manual process (Parasuraman, Sheridan, & Wickens, 2000). Despite its importance, and the variety of computational tools that exist to support it, the level of automation of conceptual design is still relatively low. Stewart's claim in the early 1990's that computational methods do a good job at solving "operational problems", but they are still far from solving real-life "messes", is still largely applicable (Stewart, 1992).

Conceptual design consists of two main processes: design synthesis and design evaluation. Forty years after the first attempts to automate the *design synthesis* process, only certain classes of well-

formulated design problems can be efficiently solved. Some instances of catalog design (Carlson-Skalak, White, & Teng, 1998) and configuration design problems (Schreiber et al., 2000) are now tractable. Very few successful applications of automatic design synthesis can be found for more open-ended problems, outside of these classes of problems.

Automating the *design evaluation* task has its own set of challenges. Thurston highlighted subjectivity as one of the biggest challenges (Thurston, 1991). The value of a design is subjective because capabilities and performance are not perceived equally by different customers. Thurston used the term *value functions* for the mapping between objective capabilities/performance and subjective value. This term is consistent with the decision analysis literature (Stewart, 1992). Other terms used to designate these functions include *preference functions* (Malen & Hancock, 1995) or *class functions* (Messac & Ismail-Yahaya, 2002). This subjectivity brings a certain degree of uncertainty to the data that the tool needs to manipulate, thus rendering any computation more complex. Techniques to handle subjective information often rely on the use of utility theory (Von Neumann & Morgenstern, 1944) or fuzzy sets (L. A. Zadeh, 1965).

Another important challenge, also identified in previous work by (Thurston, 1991) and (Malen & Hancock, 1995), is the presence of multiple and often contradicting objectives, which eliminates the uniqueness property of the underlying optimization problem (Pareto, 1896). Much has been written on methods to approach multiple criteria decision making problems. An old but still very good overview is provided in (Stewart, 1992). Two major strategies exist to tackle multiple criteria decision making: methods that transform the original problem into a single-criteria decision making problem by combining all attributes in some form, and methods that find the non-dominated set of designs.

Within the first category, value or utility-based approaches are the most common. In value-based approaches, single-attribute preferences are elicited through the use of certainty equivalents and lotteries, or pairwise comparisons. Then, they are normalized and combined through arithmetic operators, typically addition or multiplication. Multi-attribute utility theory is the most widely used of these strategies, especially in its additive form (Keeney & Raiffa, 1976). The Analytic Hierarchy Process also uses an additive value function, but the weights have different meanings and they are usually computed as the eigenvector of a reciprocal matrix containing the pairwise comparisons between all elements (Saaty, 1977, 1990). Other approaches in this first category include: a) sorting the attributes in order of priority and using a lexicographic criterion (Fishburn, 1974); b) optimizing the worst case attribute (min-max approach) (de Condorcet, 1785); c) minimizing some distance metric of the design to a "goal" or "reference" state (e.g., target performance, utopia point), such as in goal programming (Charnes & Cooper, 1957; Ignizio, 1983), or in the Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) (Lai, Liu, & Hwang, 1994).

In the second category, there exist different algorithms to find the non-dominated set: a) weighted sum methods (de Weck & Kim, 2004; L. Zadeh, 1963), where non-dominated points are obtained one by one by solving single-objective optimization problems with different combinations of weights; b) normal methods such as the normal boundary intersection method (Das & Dennis, 1998) and the normal constraint method (Messac, Ismail-Yahaya, & Mattson, 2003), which are based on the recursive search for non-dominated points in locally normal directions to the boundary of the current non-dominated set; c) constraint programming methods, in which a constraint satisfaction solver such as KodKod is used instead of an optimizer to progressively find better and better solutions; d) methods based on heuristic and meta-heuristic algorithms such as the non-dominated sorting genetic algorithm (Deb, Pratap, Agarwal, & Meyarivan, 2002), or variations such as the NSGA-PSA that increases the diversity of the approximate Pareto frontier by partitioning the frontier in clusters and picking one point from each cluster (Salomon et al., 2014). Recent work in conceptual design optimization is also exploring the definition of a set-based Pareto frontier (S-Pareto front) which defines the notion of dominance for concepts (i.e. sets of solutions) as opposed to individual solutions (Mattson & Messac, 2003).

Due to these challenges and to the superior ability of humans to holistically assess the goodness of a design, previous work has suggested keeping the human in the loop for the design evaluation process. These methods are sometimes called interactive methods. For example, the Geoffrion-Dyer-Feinberg algorithm asks the user to provide local trade-offs, in the vicinity of a solution

(Geoffrion, Dyer, & Feinberg, 1972). Kurtoglu and Campbell applied this idea to design and developed the Designer Preference Model (DPM), a tool that automatically synthesizes designs using grammar rules, presents the designs to the user for evaluation, and generates a preference model based on user assessments (Kurtoglu & Campbell, 2009). Avigad et al used a mix of objective set-based pareto ranking information and subjective concept-level human preferences to find what they call the objective-subjective front (Avigad & Moshaiov, 2009).

All in all, conceptual designers are not left without methods to choose from when approaching a particular problem. Despite the controversy that there has been around the usefulness or even the validity of some design methods, we concur with the recent editorial in this journal that scientism and praxis are complementary, not mutually exclusive, and that the research community should continue to explore both rigorous theory and tools that are seen to work empirically (Reich, 2010).

This research addresses mostly what we believe is a limitation of current tools, namely that they lack the level of effectiveness, scalability, and traceability required to be applied to large, real-life complex system architecting problems. We believe that, for conceptual design and system architecture, the best strategy to achieve these goals is to use an evolutionary optimization algorithm augmented with a knowledge-based system and direct user interaction in a model-based multi-agent framework. In this work, we focus on the use of the knowledge-based systems (KBS) to support the evaluation process. In particular, we encode expert knowledge as rules, which have been shown in the past to represent conflicting and subjective information (Bellman & Zadeh, 1970; R. R. Yager, 1977; Zimmerman, 1983).

The development of rule-based systems started with the work by Newell and Simon, who demonstrated that the way in which humans solve problems could be expressed using IF-THEN logical rules (Newell & Simon, 1972). Based on that observation, Feigenbaum and Buchanan created the first truly successful rule-based system in the field of medical diagnosis, called MYCIN (Buchanan & Shortliffe, 1984). After a great early success, work on rule-based systems stalled in the 1970's, in part due to the release of the Lighthill report (Lighthill, 1973), which gave a pessimistic view of artificial intelligence (AI) in general and rule-based expert systems (RBES) in particular, leading to funding cuts in AI labs around the world, and the subsequent "rebirth" of the field as knowledge-based systems (KBS). Lighthill argued that RBES were unlikely to ever become the world-changing technology that many though they would be. However, the development of many rule-based systems continued, with substantial success (Hart, Duda, & Einaudi, 1978; McDermott, 1982).In retrospective, RBES or KBS in general have been very successful in tackling a large variety of problems in science and engineering, and not so successful in solving general problems (Durkin, 1990). This work focuses on two goals for which RBES have shown great potential: increasing the traceability and scalability of a knowledge-intensive process such as the design evaluation process.

Note that the emphasis on traceability is not new. Work on requirements traceability abounds in the literature, especially in software systems. Spanoudakis et al used a rule-based system to generate traceability relations between documents containing requirements and use cases in natural language (Spanoudakis, Zisman, Pérez-Miñana, & Krause, 2004). Spanoudakis' work is similar to ours not only in the emphasis on traceability and the use of rules, but also in that they used rules that trace relationships between requirements and objects, and rules that trace relationships between requirements. Both these types of traceability rules are present in VASSAR, which has other rules for purposes different from traceability.

The remainder of this paper is organized as follows. In Sect. 2, a review of related work is conducted, focusing on design evaluation methodologies, and on applications of rule-based systems to other aspects of system and product development. In Sect. 3, the methodology is presented in detail using the conceptual design of a laptop as a working example. In Sect. 4, an example is presented where the methodology is applied to the conceptual design of a complex system, namely a constellation of remote sensing satellites. Finally, in Sect. 5, some conclusions and lines for future research are outlined.

2 Related Work

2.1. Classical design evaluation methodologies

Previous attempts at automating design evaluation can be classified in two broad classes: those based exclusively on objective information, and those that incorporate subjective information such as customer preferences. Objective tools include simulation tools, which are extensively used in all phases of product development: Computational Fluid Dynamics (CFD), Finite Element Models (FEM), and other ad-hoc simulation tools. Taguchi's method is another well-known example of objective design evaluation method (Taguchi, Elsayed, & Hsiang, 1989). Taguchi's method is based on the premise that product value, which is identified as *product quality*, is maximized when undesired variations in customer attributes due to variations in the manufacturing process are minimized.

Subjective methods are often based on the principles of value chain analysis (Donaldson, Ishii, & Sheppard, 2006), which essentially state that a product has value because it satisfies customer needs. Subjective methods can be further divided in qualitative and quantitative methods. Examples of qualitative methods dealing with subjective information include the Pugh matrix (Pugh, 1991) and the House of Quality (HoQ) (Hauser & Clausing, 1988). In the Pugh matrix, different design concepts are qualitatively compared to a reference design according to a set of design criteria using a 3-level grading scale (better than the reference, comparable to the reference, worse than the reference). Scores for all attributes are then simply added for each design alternative, typically assigning values of +1, 0, and -1 respectively. In the House of Quality, the effect of engineering characteristics on customer attributes is typically assessed using scores of 1, 3, and 9. The House of Quality is augmented with a variety of information about coupling between engineering characteristics, relative importance of customer attributes, and customer perceptions of the product with respect to the competition amongst others. Other qualitative approaches rely on the use of semantic qualifiers and fuzzy logic to model customer preferences, as suggested by (Park & Han, 2004) and (Gologlu & Mizrak, 2011).

Quantitative subjective tools use the methods for multi-criteria decision making described in the previous section. Multi-attribute utility theory has been extensively applied to conceptual design and architecting (Ross, Hastings, Warmkessel, & Diller, 2004). The Analytic Hierarchy Process and variants such as the Analytic Network Process are also prevalent in the concept evaluation and selection literature (Armacost, Componation, Mullens, & Swart, 1994; Dobias, 1990; Mon, Cheng, & Lin, 1994).
Finally, some design evaluation methodologies combine physical models (objective information) with customer preferences (subjective information) (Malen & Hancock, 1995). Another example of this is Ziv-av's and Reich's Subjective Objective System for generating optimal product concepts (a.k.a. SOS). SOS and VASSAR are both used to explore the space of design concepts given a predetermined set of building blocks - i.e., configuration design. Their motivation stems from two common needs: to deal with subjective and objective information to evaluate a concept, and to deal with couplings or interactions between building blocks. However, the main motivation for VASSAR, which drove the framework choice towards a knowledge-based system, was to ensure the traceability of the score, i.e. to produce a set of explanations in addition to a score behind each concept. This does not seem to be a major driver for SOS. As a result, SOS and VASSAR address their goals in fundamentally different ways. SOS is a matrix-based method, like the house of quality, design structure matrices, or what we call the Campaign-level Science Traceability Matrix (Seher, 2009), a precursor to VASSAR. Interactions between building blocks are captured in a set of adjacency matrices - in the case of SOS, one for each layer. This makes value computation efficient - value in a layer can be computed by using the layer's adjacency matrix to define an inner product in the space of designs, and the resulting quadratic optimization problem can be efficiently solved.

On the other hand, VASSAR makes the choice of sacrificing some speed to ensure traceability of the evaluation. Interactions between building blocks are modeled as individual rules with the closure property in the capabilities space, i.e., capabilities are modified or created by these "emergence rules". The effect of these new capabilities on requirement satisfaction is simulated in a subsequent step using a different set of rules, which facilitates reuse of knowledge (capabilities, emergence, and requirements) by virtue of the principle of separation of concerns.

In addition to that, SOS seems to be restricted to binary design vectors, which forces a constrained knapsack problem formulation for the optimization problem. While the concept enumeration part of VASSAR has not been the focus of this paper, VASSAR allows more flexibility in the

definition of the concept enumeration problem. Essentially, the concept space is defined by an arbitrary decision graph, where nodes are decisions of different types (knapsack, partitioning, permuting, assignment, etc) and edges indicate logical pre- and post-conditions between decisions (for example, a partitioning decision over the result of a knapsack decision).

The price to pay for this explicit simulation of emergent capabilities and traceability is of course computational time, mostly driven by the explicit simulation of emergent capabilities. However, we have also developed a methodology to make the process more efficient by pre-computing a set of N-to-1 matrices containing the interactions (capabilities + emergence) between different subsets of building blocks under different environments. These matrices are then used in real time to retrieve (instead of simulate) the emergent capabilities of a certain design. An additional layer of inter-subset emergence rules can be run at this point, before checking capabilities against requirements and producing a score.

All in all, we believe that the differences between SOS and VASSAR can be reduced to a trade-off between computational efficiency vs model expressiveness and traceability. Considering this a single-axis continuous trade-off, SOS and VASSAR would be on different parts of the spectrum, and therefore they can be seen as complementary tools.

In general, most design methodologies work well for design problems that use knowledge bases of limited size. However, real life conceptual design problems are becoming more and more knowledge-intensive: nowadays, it would not be unusual for a design tool to require large databases containing thousands of customer requirements (Wnuk, Regnell, & Schrewelius, 2009), or hundreds of engineering models, for a complex system. This fact raises two problems: a) it becomes hard to add or modify knowledge to these knowledge bases; b) it becomes hard to trace the results of the tool (i.e., the value of the design) back to the driving customer requirements and engineering characteristics.

This work proposes an architecture evaluation tool that leverages the natural *traceability* and *scalability* of rule-based systems to alleviate these two problems. The rules engine also facilitates the use of fuzzy sets to model customer preferences, and the use of simple rules to model complex emergent behavior (Wolfram, 2002) that is often at the origin of value delivery (Crawley et al., 2004).

2.2. Knowledge-based design methods

VASSAR uses a rule-based system, but many alternative methods and architectures exist for expert systems. A relatively recent survey of expert systems methods between 1995 and 2004 identified ten expert systems methods, including rule-based systems, neural networks, fuzzy expert systems, object-oriented approaches (including frame-based systems), case-based reasoning, intelligent agents, ontologies, and databases among others (Liao, 2005).

Multiple examples can also be found that combine several methods. For instance, Bonczek et al combine first-order logic with database techniques to enhance decision support (Bonczek, Holsapple, & Whinston, 1981).

In complex knowledge-based systems, knowledge is organized in clusters that are often called modules or agents. For example, when designing a satellite, an agent can take on the role of designer of one of the subsystems, such as propulsion or attitude determination and control. Similarly, VASSAR has different types of rules for computing capabilities, assessing requirement satisfaction, and simulating emergent behavior, all of which can be seen as independent agents.

Different multi-agent architectures differ in their way of handling communication between agents. A traditional way of communication between agents is directly connected architectures, where connections between agents are static and predefined in a data flow diagram. Examples of this paradigm can be found in the CommonKADS knowledge templates (Schreiber et al., 2000). CommonKADS has a template for configuration design that consists in three main steps executed iteratively: propose-verify-revise. CommonKADS also contains a template for a general assessment task, but there is no template for design evaluation.

Indirectly connected architectures such as blackboard architectures appeared in the 1970's with systems such as Hearsay-II (Erman & Hayes-Roth, 1980), and flourished (Engelmore & Morgan, 1988) in part as a reaction to the rigidity of directly connected architectures. In blackboard architectures, there are several independent knowledge sources that know the conditions under which they can contribute to the problem (trigger conditions) and how to solve a specific part of it. There is also a central repository, the blackboard, that acts as common interface for all modules to communicate. Finally, there is a central moderator that decides which path to choose among all possible paths (Nii, 1986). Blackboard architectures were considered by Feigenbaum as the most flexible architecture for a knowledge-based system (Engelmore & Morgan, 1988), since the blackboard acts as a common interface that enables the independent evolution of knowledge sources.

Blackboard architectures are conceptually similar to rule-based systems: the working memory could be the blackboard; the rules could be the knowledge sources. However, knowledge sources are typically more complex than single rules in rule-based systems, closer to modules in rule-based systems. In other words, a knowledge source could itself be implemented as a rule-based system (Corkill, 2003). Thus, blackboard architectures go one step further in the direction of flexibility than rule-based systems. An example of a blackboard system applied to design is Fenves et al's knowledge-based system for automatic control of standards of structural design in CAD programs (Fenves & Garrett, 1986). Fenves et al's use of rules for standard verification is similar to our requirement rules, but the intent is quite different: Fenves used a Boolean assessment of standard satisfaction, whereas we provide a quantitative or semi-quantitative evaluation for individual requirements, and a combination of weighted averages and logical operators for aggregation of requirement satisfaction.

More recently, the emphasis has been on multi-agent architectures such as MadKit's Agent-Group-Role (Gutknecht & Ferber, 2001). In MadKit, agents can communicate directly with each other without going through a central repository. Corkill proposed that blackboard architectures are more appropriate than agent-based architectures when agents are to collaborate closely, due to the unnecessary performance penalty paid by agent interaction, whereas multi-agent architectures are more appropriate for large, distributed applications with long-lived, loosely coupled agents (Corkill, 2003). An example of design tool with an agent-based architecture is A-design (Campbell et al., 1999).

2.3. Rule-based systems for design

The idea of using logical rules in computer programs to model the expert problem solving process was first developed by Ed Feigenbaum at Stanford (Feigenbaum, Buchanan, & Lederberg, 1971; Lindsay, Buchanan, & Feigenbaum, 1993), based on findings by Carnegie Mellon's cognitive psychologists Allen Newell and Herbert Simon that expert knowledge is best modeled in chunks that can be represented in the form of if-then statements (Newell & Simon, 1972). Since then, rule-based systems have been applied to virtually every discipline of science and engineering (Clancey, 1987; Dincbas, 1980; Duda, Gaschnig, & Hart, 1979; Durkin, 1990; McDermott, 1982).

The use of rule-based systems is not new in formal design methodologies. Different forms of rule-based systems have been used for design synthesis (Antonsson & Cagan, 2001; Stiny, 1980). In design evaluation, rule-based systems have been used in the past for some aspects of both value assessment (Gologlu & Mizrak, 2011; Park & Han, 2004), and cost assessment (Mauchand, Siadat, Bernard, & Perry, 2008).

This work proposes a common rule-based methodology for both value and cost assessment. The framework bridges previous work by (Gologlu & Mizrak, 2011; Mauchand et al., 2008; Thurston, 1991) by encoding both performance evaluation using rules and cost assessment as rules, and pushes forward in the direction of increased scalability and traceability of architecture evaluation tools, in order to facilitate the use of these tools for large, knowledge-intensive design problems.

3 A Rule-Based Method for Scalable and Traceable Design Evaluation

This section introduces a rule-based methodology for a scalable, traceable, and customer-centric conceptual design evaluation tool. We call this methodology VASSAR (Value ASsessment of

System Architectures using Rules)[1]. The VASSAR methodology leverages the communicative power of logical rules as data structures, the natural recursivity of functional languages, and the traceability and scalability of rules engines, to efficiently incorporate a large quantity of expert knowledge into conceptual design evaluation. A laptop design problem is used as a simple example to illustrate the methodology. The methodology is then applied to the large-scale problem of a constellation of Earth Observing satellites in Sect. 4.

3.1. Applicability

It is important to note that the VASSAR methodology was developed to be used in a typical system architecture framework, where the elements of the system architecture and their relationships are sampled from a finite and predetermined set of alternatives. This limitation arises from the fact that rules need to be created linking architectural elements to capabilities and performance. System architecture problems are closest in nature to design configuration problems where all design variables are discrete or categorical. These problems usually involve optimization over spaces of subsets, partitions, permutations, matchings, and more generally any kind of graph. An in-depth discussion of the classes of architecture problems that can be tackled with VASSAR is provided in (Selva, 2012) (chapter 2). Thus, VASSAR is not applicable to other types of design problems, especially those involving a large number of continuous variables, such as shape, structure, or aerodynamic optimization problems.

3.2. Data Structures

Rule-based systems use two fundamental data structures to store data in *working memory*, namely *facts* and *rules*. Facts are lists of unordered pairs *(slot value)* that contain information about the problem at hand. Rules consist of a left-hand side (LHS) containing a set of conditional elements, and a right-hand side (RHS) containing a set of actions. A rule-based system works by matching all facts to the LHS of rules in working memory and creating activation records for each match. At each iteration, the inference engine takes all current activations, decides which rule to *fire* next, and executes the actions in the RHS of this rule. Typically, actions on the RHS include *asserting* other facts, i.e., adding new facts into working memory, which then match other rules.

In addition to facts and rules, VASSAR uses *fuzzy numbers* to handle inexact reasoning in both customer attributes and engineering characteristics. Fuzzy numbers are easier to elicit from experts than traditional utility functions. Our fuzzy numbers are simple implementations of Zadeh's fuzzy sets that use triangular membership functions and center-of-gravity "defuzzyfication". For example, the level at which a certain customer attribute is satisfied is encoded using a fuzzy number, according to the membership functions illustrated in Fig. 1. In addition to these fuzzy numbers, crisp values of 0% and 100% satisfaction are also possible for extreme cases in which there is no uncertainty concerning the satisfaction of the requirement. This is useful to encode "utopia" and "show-stopper"' scenarios.

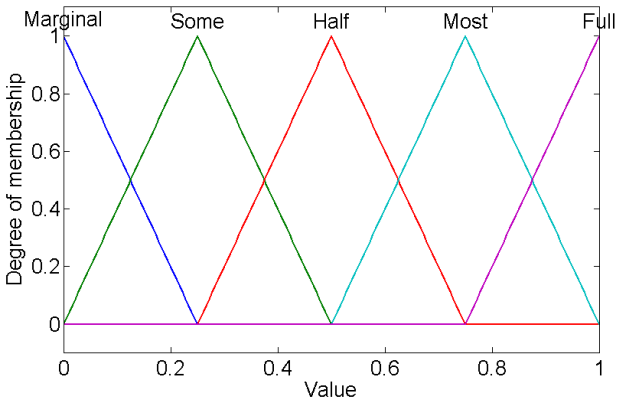


Fig. 1: Membership functions used for customer attributes

[1] An earlier version of the VASSAR methodology was presented at the 2013 IEEE Aerospace Conference (Selva & Crawley, 2013), although that paper focused on the implications for space system architecting.

A conceptual design or architecture is represented as a fact where slots represent design decisions. An example of a fact containing the conceptual design of a laptop in a catalog design environment is provided below.

Laptop Architecture Fact
Id: Laptop-1
Processor: QUAD-2.2GHz-64bit-6M
RAM: 16GB-DDR3-1600MHz
Hard-disk: SSD-256GB
Optical-drive: HDD-1TB-7200RPM
Graphics Unit: 2GB-GDDR5-128bit
Battery: 9C-65WHr
Screen: FHD-15in-COLORSEN

In this example, a laptop is defined eight attributes, one of which is an identification string.

Customer satisfaction depends on the capabilities and performance of the design. In the context of this paper, we refer to performance as the attributes of a capability. For instance, gaming laptops all have the capability of playing resource-consuming video-games, but different gaming laptops may fulfill this capability at difference performance levels. Capabilities and performance are represented by capability facts. For example, a laptop "portability" capability fact illustrating several performance attributes is given below. Note that attributes can be numerical (e.g., autonomy is 6.5 hrs), or fuzzy (e.g., weight is "Heavy").

Laptop Capability (Portability) Fact
Id: Laptop-1
Autonomy-hrs: 6.5
Weight: "Heavy"

Customer preferences are formally captured through requirements. A requirement is represented as one or more rules whose LHS matches a certain combination of capabilities facts, and whose RHS asserts a requirement satisfaction fact containing the corresponding level of satisfaction. Note that requirements can match both numerical and fuzzy attributes. In the case of the laptop, a requirement related to portability might be defined by the following rule:

Laptop Requirement Satisfaction Rule
LHS: Match *Laptop Capability (Portability) fact* with:
Id: Laptop-1
Autonomy-hrs: At least 4.5hrs
Weight: "Light" or better (e.g., "Ultra-light")

RHS: Assert *Requirement Satisfaction fact* with:
Requirement id: REQ1-1
Satisfaction level: "Full"
Satisfied by: *arch1*.

3.3. First-level decomposition of VASSAR

Customer-centric design evaluation is based on the premise that the value of a conceptual design should capture the ability of that design to meet customer needs. VASSAR takes this approach and assesses the value of a concept or architecture in three steps: a) compute the capabilities and performance of the architecture using engineering models; b) translate capabilities and performance to preference by comparing them with customer requirements; c) aggregate requirement satisfaction into a handful of metrics. In parallel of these three processes, an *explanation facility* keeps track of the value chain, and constructs a set of explanations that are provided to the user with the value fuzzy number. This process is illustrated in Fig. 2, where boxes indicate objects, ovals indicate processes, and arrows illustrate the dependencies between them (e.g., the overall flow of information). The circle connector indicates an enabler (e.g., the domain knowledge base enables the assertion of the capabilities of the architecture).

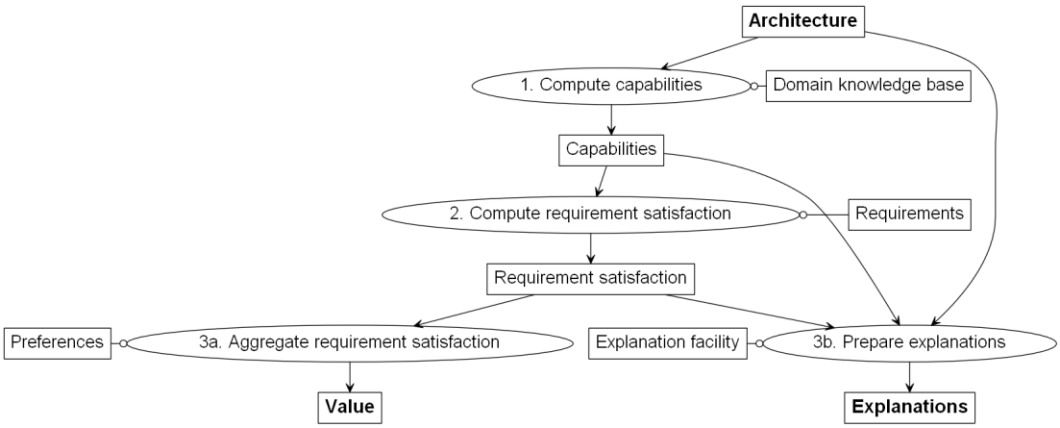


Fig. 2: First-level decomposition of the VASSAR methodology

3.4. Level-2 model of VASSAR

In this subsection, the four processes of the VASSAR methodology are described in more detail.

3.4.1. Step 1: Compute Capabilities

Capabilities can be computed from design decisions using domain knowledge in the form, for example, of physics laws or engineering models. The process of computing capabilities can be decomposed into five main processes, as illustrated in Fig. 3: a) asserting architectural facts; b) inheriting attributes from upper to lower levels of the hierarchy of form decomposition; c) computing basic capabilities; d) computing performance; e) computing emergent capabilities. Each of these steps is described in more detail below. Double arrows on Fig. 3 represent situations in which a process modifies an existing set of facts, as opposed to creating a new set of facts.

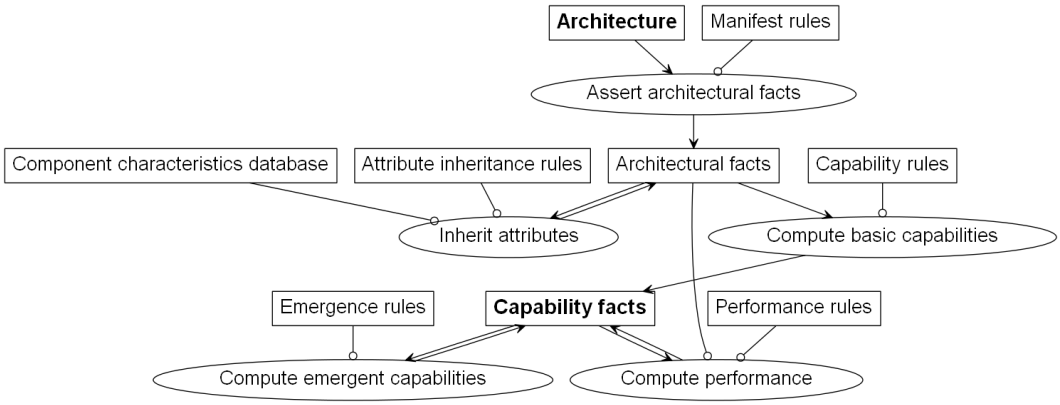


Fig. 3: Zoom-in on Step 1: Compute capabilities

a) Asserting architectural facts: First, the process takes a conceptual design or architecture and asserts all the corresponding system, subsystem, and component facts using a set of rules labeled as *manifest rules*. The current version of VASSAR is limited to a 3-level form decomposition nomenclature (system, subsystems, and components) following INCOSE's recommendation (Haskins, 2006). Two levels may be sufficient for simple conceptual design problems. For example, in the case of a laptop, the subsystem level is probably unnecessary: the system is the laptop, and examples of components are the hard-disk and the processor.

b) Inheriting attributes: Subsystem and component facts created through manifest rules typically have many slots (attributes) empty, because not all properties are directly set by design variables. For example, given an architecture such as arch1 that has a QUAD-2.2GHz-64bit-6M processor, a component fact will be asserted by manifest rules that has the Id field set to QUAD-2.2GHz-64bit-6M, and the rest of attributes (e.g., CPU speed, cache size, number of bits) empty. These attributes are inherited in the second step by *attribute inheritance rules*. There are two main types of attribute inheritance: a) direct inheritance from upper levels of the architectural hierarchy, e.g., the architecture id property is inherited from the architecture fact to the subsystem or component facts; b) direct inheritance from a database of fixed parameters such as components characteristics, e.g., the amount of memory of a video card is imported from a database of video cards.

c) Computing basic capabilities: The third step is the assertion of an initial set of capability facts through the application of *capability rules*. These capability rules have the following structure: "IF there is a certain combination of components with certain attributes, THEN assert a certain capability." For example, a rule asserting the capability of a laptop to play video-games is provided below.

Laptop Capability Rule

LHS: Match *Laptop Architecture Component (RAM) fact* with:
Parent: *arch1*
Id: Id_Comp_RAM
Memory-GB: Larger than 8

Match Laptop Architecture Component (Graphics Unit) fact with:
Parent: *arch1*
Id: Id_Comp_GPU
Memory-GB: Larger than 2

Match Laptop Architecture Component (Screen) fact with:
Parent: *arch1*
Id: Id_ Comp_Screen
Size-in: At least 17

RHS: Assert *Capability (Gaming) fact* with:
Parent: *arch1*.
By: combination of *Id_ Comp_RAM, Id_ Comp_GPU, Id_ Comp_Screen.*

d) Computing performance: *Performance rules* fill out attributes of capability facts (e.g., portability, gaming), typically by combining information from different architectural facts (e.g. RAM, graphics unit). For example, the weight of the laptop is directly copied from the *Laptop Architecture* fact to the *Portability Capability* fact, and the expected performance of a laptop in a complex simulation depends on several attributes from the *Processor Component* fact (speed), the *RAM Component Fact* (total memory, speed) and the *Hard-disk Component fact* (speed). Performance rules only fire when all the attributes that they need for the computation are available, i.e., when they have been inherited or computed by other performance rules.

e) Computing emergent capabilities: *Emergence rules* are responsible for asserting new capabilities from combinations of existing capabilities. Their structure is the following: "IF there is a certain combination of capabilities with certain performance, THEN assert a new capability with a new performance". An example of emergence rule is provided below.

Laptop Emergence Rule

LHS: Match *Capability (Portability) fact* with:
Parent: *arch1*.
Id: *Id_Capa_Porta*
Autonomy-hrs: More than 4h
Weight: "Light" or Better

Match Laptop Capability (Touch-Screen) fact with:
Parent: *arch1*
Id: Id_Capa_Touch
Writing-recognition: Yes
Screen-fold-down: Yes

RHS: Assert *Capability (Class-Notes-Taking) fact* with:
Parent: *arch1*.
By: combination of *Id_ Capa_Porta, Id_Capa_Touch.*

Note that the two last processes of computing emergent capabilities and performance do not occur sequentially: the performance of both basic and emergent capabilities is computed as the required

attributes become available; emergent capabilities are asserted as their required capabilities with the required performance attributes are asserted; this runs iteratively in a loop until no more emergent capabilities can be asserted, and all performance attributes that can be calculated have been calculated.

3.4.2. Step 2: Compute Requirement Satisfaction

Thurston and Messac amongst others noted that capabilities do not linearly translate into value (Messac & Ismail-Yahaya, 2002; Thurston, 1991). For example, the benefit of more computational power in a laptop does not increase linearly with computational power. Instead, the relationship between capabilities and requirement satisfaction may exhibit discrete steps (e.g., the ability to run a specific program), saturation effects (e.g., we get no marginal benefit from adding more computational power after a certain level because we are limited by the performance of other elements such as the operating system), and non-linear continuous multiplicative envelopes (e.g., a concave envelope capturing decreasing marginal benefit). These effects are captured by functions that map performance to satisfaction. These functions have received different names in the past, namely *value functions* (Thurston, 1991), *class functions* (Messac & Ismail-Yahaya, 2002), or *preference functions* (Malen & Hancock, 1995). From now on we adopt Malen's nomenclature and refer to them as preference functions. In the VASSAR methodology, preference functions are expressed in the form of requirement satisfaction rules that capture these discrete steps, saturation effects, and non-linear continuous envelopes. Requirement satisfaction rules look for combinations of capability facts and assert requirement satisfaction facts, as illustrated in Fig. 4. Section 4 delves deeper into different approaches to model preferences using rules.

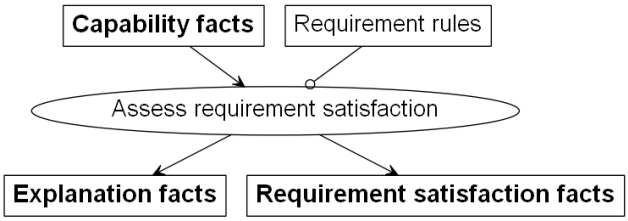


Fig. 4: Zoom-in on Step 2: Compute preferences

3.4.3. Step 3: Aggregate Requirement Satisfaction

The preference domain is hyperdimensional because there is at least one dimension per requirement, and a complex system can easily contain several hundreds or even thousands of requirements. Dominated architectures can easily be eliminated by means of Pareto analysis, but this is unlikely to reduce the size of the tradespace by much because, in general, the size of the non-dominant set increases with the number of metrics. In particular, full satisfaction of one requirement is sufficient to make an architecture non-dominated. Hence, the problem remains of choosing between non-dominated architectures that have very different requirement satisfaction sets. In other words, the dimensionality of this domain needs to be reduced in order to be able to make a decision concerning the preferred architectures. This requires the use of subjective information capturing customer preferences.

Hence, the last step of the methodology is the reduction of the dimensionality of the preference domain through aggregation of requirement satisfaction into objective satisfaction, and then into customer or more generally user satisfaction. Again, a 3-level hierarchy is assumed for satisfaction (user needs, objectives, and requirements). This two-step aggregation process requires the elicitation of a list of objectives and requirements from different users, and reduces the dimensionality of the preference domain (i.e., the number of metrics) from the number of requirements to the number of users. For example, in the case of the laptop, one could imagine a family with three members with conflicting requirements, who only has resources to buy one high performance laptop or perhaps two lower performance laptops, and therefore need to come up with an optimal conceptual design for their computation needs. In this case, a possible hierarchy of users-objective-requirements is illustrated below.

Mathematically, requirement aggregation rules reduce the dimensionality of the satisfaction space by using *aggregation functions*. Aggregation functions combine arithmetic and logical operators. The simplest and most commonly used arithmetic operator is the weighted average. For example, the satisfaction of a user is given by a weighted average of the satisfaction of its objectives. More

sophisticated arithmetic operators are also possible (see for example Yager's ordered weighted averaging operator, or Fortin's gradual numbers (Fortin, Dubois, & Fargier, 2008; R. Yager, 1988)). Logical operators can also be utilized to express preferences of the type, such as the *at-least-n-out-of-k* condition: "*the stakeholder is satisfied if 3 or more of their 4 objectives are satisfied.*"

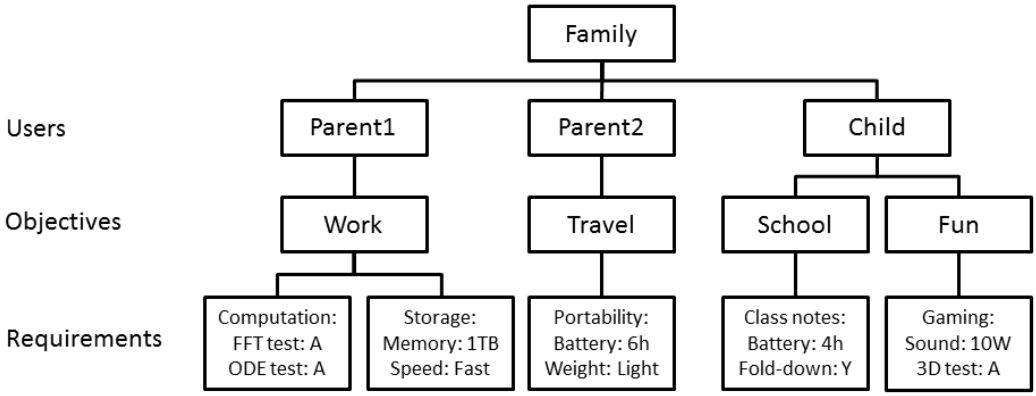


Fig. 5: Hierarchy showing customers, objectives, and requirements for the laptop example

The number of metrics can be further reduced to 1 if the relative importance of users to a central stakeholder is introduced. The relative importance of users (e.g., Parent1, Parent2, Child) can be formally computed by applying quantitative stakeholder analysis techniques, such as Cameron and Crawley's stakeholder networks (Cameron, 2008). If this third aggregation step is applied, the fuzzy number that is obtained represents the value of the system architecture to the central stakeholder. The entire process is illustrated in Fig. 6.

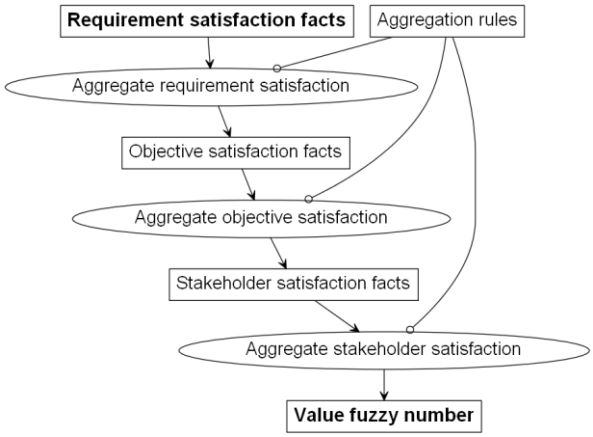


Fig. 6: Zoom-in on Step 3a: Aggregate preferences

3.4.4. Step 4: Prepare Explanations

The VASSAR methodology provides a set of explanations that accompany the fuzzy value metric. This is central to the methodology, as it satisfies one of its key requirements, namely that of showing traceability of the value chain.

Explanation rules keep track of the entire value chain by extracting information from architectural facts, capability facts, requirement satisfaction facts, objective satisfaction facts, and stakeholder satisfaction facts, as illustrated in Fig. 7.

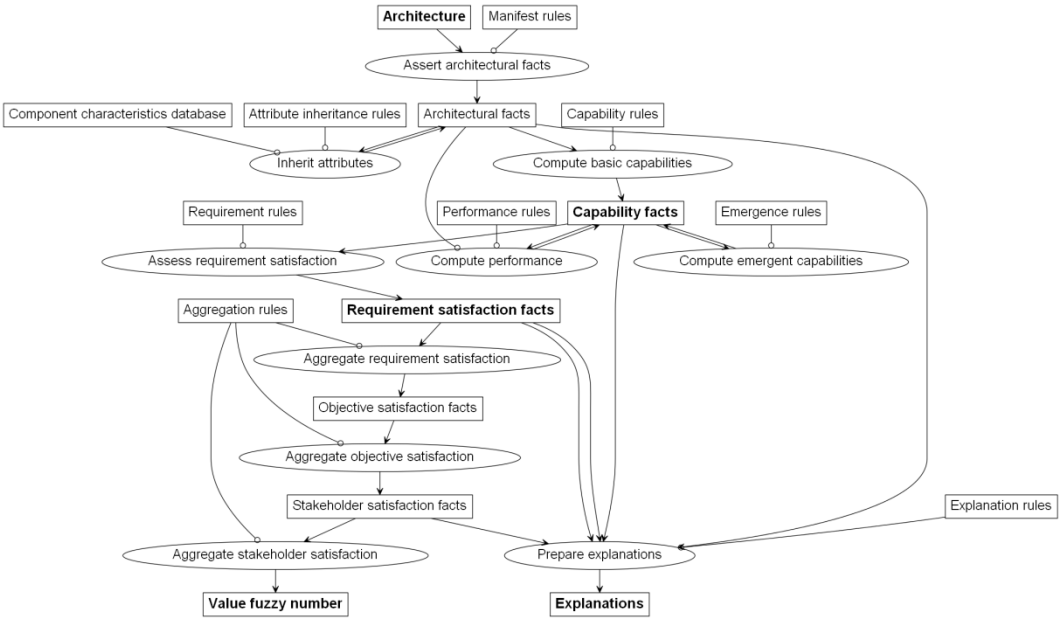


Fig. 7: The explanation facility extracts information from different points of the process and prepares explanations that accompany the results.

3.5. Some Notes on Preference Functions and Requirement Satisfaction Rules

Previous work by Messac amongst others has looked at different forms for preference functions (Messac & Ismail-Yahaya, 2002). In VASSAR, preference functions are embedded in requirement satisfaction rules. In the following paragraphs we discuss different implementations for requirement satisfaction rules for these different types of preference functions. We start by discussing single-attribute preference functions, and then discuss different strategies to generalize to multiple attributes.

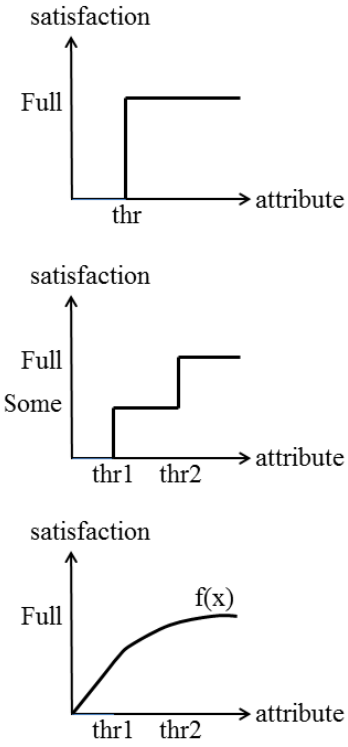
3.5.1. Implementation of Rule-Based Single-attribute Preference Functions

The simplest possible single-attribute preference function is an identity function. This implies a perfectly linear relationship between capabilities and satisfaction. For example, one could argue that the preference function for the weight of a laptop is linear, so that Laptop 1 weighing half as much as another Laptop 2 provides twice as much value to its customers. However, this approach is often too simplistic in practice and is rarely useful.

Single-step preference functions adequately capture cases in which preferences are driven by a single threshold value. For example, two requirements on the portability of a laptop may read "the laptop shall weigh less than 3 lb", and "its maximum dimension shall not exceed 12 in". This type of requirements can be modeled as step functions, where value is zero below the threshold level and 1 at or beyond the threshold level. Single-step preference functions are very simple to implement using one rule per requirement (see Fig. 8). Note in particular that it is easy to include a short sentence providing the justification behind this threshold. This information can readily be used by an explanation facility to explain to the user why a certain requirement is satisfied. Moreover, many requirements in practice consist of a single threshold value. One threshold is however not sufficient to capture complex preference functions.

An arbitrary *continuous preference function* can capture any subtlety in the mapping of the multi-dimensional performance attribute space to the one-dimensional requirement satisfaction space. Continuous preference functions are also easy to model with one rule, where the performance attribute is matched in the left-hand side of the rule (LHS) and the continuous function is used in the right-hand side of the rule (RHS) to transform to value (see Fig. 8). However, continuous satisfaction functions have two disadvantages: a) their development requires eliciting large amounts of information for stakeholder requirement; b) they are less suited to directly feed information to an explanation facility. Indeed, the explanation facility would have to infer local information from the overall global information about the shape of the continuous preference function.

Multi-step preference functions, defined by a set of thresholds and a set of corresponding preference values, are a good compromise between single-step functions and arbitrary continuous functions. First, many system requirement documents express requirements using a target value and a threshold value, which suggests the use of bi-step preference functions (N=2). Second, they are easy to implement by using one rule per threshold. Third, maintaining one rule per threshold maintains the suitability for the explanation facility. And finally, continuous preference functions can be approximated by multi-step preference functions to an arbitrary degree of fidelity by increasing the number of thresholds. In particular, any of Messac's class functions can be modeled (Messac & Ismail-Yahaya, 2002). From our experience modeling six different aerospace systems in this framework, we have found that in practice, it is very rare that more than five thresholds are needed to capture the main features of a preference function.



Single Threshold Requirement Rule
LHS: Match *Capability fact* with:
Att1: Greater than *thr*.
RHS: Assert Satisfaction fact
Satisfaction level: "Full"

Multiple Threshold Requirement Rules

LHS: Match *Capability fact* with:
Att1: Between *thr1 and thr2*.
RHS: Assert Satisfaction fact
Satisfaction level: "Some"

LHS: Match *Capability fact* with:
Att1: Greater than *thr1*
RHS: Assert Satisfaction fact
Satisfaction level: "Full"

Continuous Function Requirement Rule
LHS: Match *Capability fact* with:
Att1: x.
RHS: Assert Satisfaction fact
Satisfaction level: F(x)

Fig. 8: Rule-based implementation of single-step (upper), multi-step (center), and continuous (lower) single-attribute preference functions

3.5.2. Extension to Multiple Attributes

One approach to extend this analysis to multiple attributes is to simply use single-attribute preference functions and perform the combination at the satisfaction level. For example, in the preceding example, two different rules and preference functions could be used for weight and maximum dimension. This is equivalent to deferring the treatment of multiple attributes to Step 3 (Aggregating preferences). However, this approach assumes some degree of independence between attributes. In reality, this might be an incorrect assumption, because the preference function of an attribute may depend on the value of another attribute. For example, the threshold for weight may depend on the maximum dimension of the system, and vice-versa. Additional rules would be required to handle these couplings.

Instead, multi-dimensional preference functions can be used to take into account several attributes simultaneously. The drawback of this approach is that it could potentially require many rules to cover all cases, i.e., all possible combinations of values of the attributes. For example, assuming a 3-level preference function for weight and maximum dimension leads to 9 possible performance scenarios. A multi-dimensional preference function would thus require in principle 9 rules, one per scenario. The one-dimensional preference function would require 2 rules (one per attribute), plus as many rules as needed in order to account for all the trade-offs between attributes.

An important remark concerning this methodology is that one rule covers one particular case (e.g., the nominal case in which weight and maximum dimension are all at or above the desired level for those attributes). If any of these conditions is not met (i.e., if any of the attributes is below the desired level), this rule will not fire. If no other rules expressing full or partial satisfaction of this requirement are fired, it will be considered that the architecture provides a null (0%) satisfaction of that requirement. Therefore, it is important to cover all necessary cases for each requirement.

One could think that this would lead to an infeasible number of requirement satisfaction rules capturing all possible degraded cases. In practice however, we have found from conversations with experts that one rule describing the conditions for full satisfaction plus a handful of partial satisfaction rules can cover all realistic cases for a given set of attributes.

In practice, the choice between single-attribute preference functions and multi-dimensional preference functions is problem-specific and depends on factors such as the number and coupling between attributes.

3.6. Comparison with other rule-based evaluation frameworks

We mentioned in the literature review that this is not the first attempt to use knowledge-based systems to support the conceptual design phase. We believe, however, that the rule-based VASSAR architecture with capability, requirement, emergence, aggregation, and explanation rules, and macros that write rules based on the contents of a database facilitates scalability and traceability. This is discussed further in this section.

3.6.1. Scalability

Scalability of rule-based systems in the sense of ease of change comes mainly from the fact that the expert knowledge is physically separated from the rest of the code, so it can be modified and updated independently of the rest of the code. Adding more rules is as simple as loading another text file containing the additional rules. No other steps are necessary as long as: a) all the modules and templates used in the new rules are defined in the system; b) the new rules correctly specify their pre-conditions with respect to other rules. The organization of the rules in different modules further facilitates this task.

The scalability of our tool in particular is also supported by an automatic programming module that reads input information from a spreadsheet and automatically writes rules of different types. For example, in one particular project, we automatically generated 2,000+ requirement rules by reading the requirements from a spreadsheet.

Scalability can also be understood as the ability to handle very large problems, which is related to performance. The performance of a naïve rule-based system is O(RF^P), where R is the number of rules, F is the number of facts, and P is the average number of patterns in the rules. However, the performance of Jess, which uses an improved version of the Rete algorithm, is closer to O(RFP), so it grows linearly with the size of the working memory. This increases scalability because the user can arguably tackle larger problems with a pure rule-based system architecture than with a multi-agent architecture that has overhead for handling communication and coordination between agents.

Furthermore, both the Jess rule-based system and the framework presented are very easy to incorporate in a parallel computation environment, which we have done in all our projects. Multiple Rete objects can be created, initialized with the rules, and then used by any number of workers to evaluate architectures in parallel (the object just needs to be reset every time it receives a new architecture).

3.6.2. Traceability

It is often mentioned in textbooks that rule-based systems offer natural traceability, since the simple sequence of rules that were executed traces the reasoning process of the system (Giarratano & Riley, 2004).

However, the key to the traceability of VASSAR comes from an ad-hoc module, namely explanation rules. Explanation rules keep track of all the important - and only the important - events that occur during execution, and generate explanation facts with that information. These

explanation facts are later used by an explanation facility to provide both graphical and text-based explanations to the user on demand.

For example, the user can ask: a) to see how a certain capability and performance emerges from the interaction of subsystem and components interfaces and characteristics; b) to see the traceability of how (i.e., by which combination of capabilities) a particular requirement was satisfied by a certain design; c) to see how an overall score decomposes into stakeholder, objective and requirement satisfaction.

Ultimately, this traceability increases the confidence the user has on the tool.

3.7. Validation

Verification and validation (V&V) of a decision support tool is challenging because there is rarely a truth value with which the output of the tool can be compared. Still, V&V is a critical aspect of any decision support tool, and substantial effort was put into the V&V of the VASSAR framework.

In V&V, one typically differentiates between validation (the tool does the right thing) and verification (the tool does what it is supposed to do correctly). Verification of the tool was largely done by comparison to other automatic decision support tools based on general purpose frameworks, namely Matlab and in some aspects Excel.

Validation of decision support tools is always challenging, especially when the tool is applied to a current problem for which the solution is not known. The approach that we took for validation is two-fold. First, we applied the tool to a well-known retrospective case study, namely the NASA Earth Observing System designed in the 1980's. The results that we found were compared to the literature and reviewed in detail with a senior manager at NASA that was in charge of developing that program in the 1980's. Results were found to be consistent with the literature, and the reviewer also found the results consistent.

To give a specific example, the tool predicted the assignment of correctly predicted the assignment of 12 instruments to spacecraft based on measurement synergies, conflicts, and cost, with only one difference from the chosen assignment. Specifically, the tool predicted that a certain atmospheric chemistry instrument in the EOS program should fly in a different spacecraft than it actually did. The tool identified that this one instrument would increase science return through synergies between instruments and would decrease cost through a down-grade in one of the launch vehicles required. Our expert confirmed these arguments, and explained that the instrument was in fact originally slated to fly as described in the tool (information that was not publicly available). The instrument in question was an international instrument, and there was a hard constraint at the time to fly that instrument in the first spacecraft, even though that was a suboptimal decision in terms of science and cost. (All this is explained in detail in (Selva, 2012), chapter 6).
The framework has also been used for projects in communications satellites. In this case, results have been presented and discussed in detail with NASA personnel, and they have always found them satisfactory in terms of fidelity. We found that the traceability feature of the tool plays a crucial role in helping users gain trust in its ability to generate useful results.

While the tool has been mostly applied to problems in the aerospace domain, we are confident that it is useful for any system architecture problem that can be expressed by means of a decision graph following the description provided in Section 3.1.

4 Application to a Knowledge-Intensive Architecting Problem: A Constellation of Soil Moisture Monitoring Satellites

We emphasized earlier the importance of the applicability of a design evaluation tool to knowledge-intensive problems. In this section, we apply VASSAR to a soil moisture remote sensing satellite mission. This hypothetical mission would combine instruments that are similar to real instruments flown or to be flown in American and European missions. This example was designed to highlight some of the difficulties that systems engineers find to apply design evaluation methodologies, and to illustrate different features of VASSAR. First, it is a knowledge-intensive problem, as a large body of expert knowledge is required to assess the worth of an architecture (e.g., hydrology, climatology, numerical weather prediction, satellite remote sensing,

and so forth). Second, a large portion of the value delivered by such a mission is scientific value, which contains elements of subjectivity and is hard to model in traditional design evaluation methodologies presented in Sect. 2. Finally, this system has a complex value chain loop with clear examples of emergent behavior that actually drives an important part of the value delivery. This will test the ability of VASSAR to provide traceability of the value chain.

4.1. Instrument description

Five instruments are considered in this example, namely: a) an L-band polarimetric radiometer that we call LRADIO, based on the design of NASA's SMAP mission (Entekhabi, 2010); b) an L-band synthetic aperture radar (LSAR), also based on an instrument on the SMAP mission (Entekhabi, 2010); c) an X-band polarimetric radiometer (XRADIO), based on NPOESS/CMIS (Gasster & Flaming, 1998); d) an infrared multispectral radiometer (IR) based on NPOESS/VIIRS (Welsch & Swenson, 2001); e) a hypothetical P-band polarimetric synthetic aperture radar (PSAR), such as the one proposed for BIOMASS (Heliere et al., 2009). The characteristics (mass, power, data rate, performance) that we assumed for these instruments are provided in the Appendix.

4.2. Architecture tradespace

Although the work presented in this paper focuses on design evaluation, this section presents a complete tradespace exploration problem including enumeration and evaluation of a finite set of architectures. For this example, we chose to represent an architecture as the set of decisions laid out in Table 1. The set of possible architectures is given by:

$$N_{arch} = 3 \times 3 \times 2 \times \sum_{n=1}^{5} \binom{5}{n} Bell(n) = 3{,}636$$

where N_{arch} is the number of possible architectures, and $Bell(n)$ is the nth Bell number.

Table 1: Architectural decisions and range of values

Decision	Range of values
Payload selection	Any combination of the 5 instruments described in the payload description subsection (31 combinations excluding the empty set)
Payload-to-spacecraft allocation	Any partition of the payload set into spacecraft (between 1 and 52 combinations, depending on N_{instr})
#satellites per plane	[1,2]
Orbit altitude	{400;600;800}
Orbit type	{polar; sun-synchronous (SSO) dawn-dusk (DD); SSO morning (AM)}

4.3. Stakeholder requirements

Five stakeholder groups or panels were identified for this example, labeled as follows: weather, climate, ecosystems, water, and applications. All five panels were initially considered equally important; a case with non-uniform weights is considered later in the sensitivity analysis. Note that the panels' relative weights could be obtained through a formal method such as the one proposed by Cameron and Crawley (Cameron, 2008). The specific objectives of each panel, as well as their relative ranking, are provided in the Appendix.

4.4. Capability rules

Nominal instrument capabilities are presented in the Appendix. However, architectural decisions can affect these instrument capabilities in non-trivial ways that are encoded in logical rules. For example, temporal resolution and spatial resolution both depend on the orbital parameters. Data quality also depends on the orbital parameters. Below are a few examples of situations in which data quality is severely compromised due to orbital parameters.

- Lighting conditions: Visible and near-infrared instruments (namely the corresponding channels of the infrared radiometer) cannot work in sun-synchronous orbits with pre-dusk local times of the ascending/descending node because they cannot gather enough light.

- Image distortion: Side-looking instruments cannot work at low altitudes because image distortion becomes unacceptable.

4.5 Performance rules

Performance rules were added in this example to compute measurement attributes such as spatial and temporal resolution from instrument characteristics and orbital parameters. For example, the spatial resolution Δx of a side-looking microwave imager of aperture D and frequency f with an off-nadir angle of θ flying at an altitude h can be approximated by the following rule-of-thumb

$$\Delta x = 2h\left(\tan\left(\theta + \frac{c}{2fD}\right) - \tan\left(\theta - \frac{c}{2fD}\right)\right)$$

4.6. Emergence rules

Emergent behavior plays a key role in value delivery to stakeholders. We describe in this subsection a few examples of emergent behavior (both in science and cost) that were implemented in this case study.

- Data disaggregation schemes: The high accuracy, low spatial resolution soil moisture dataset provided by LRADIO can be combined with the lower accuracy, higher spatial resolution dataset provided by LSAR to produce a new high accuracy, medium spatial resolution dataset.

- Sample averaging: In any dataset, part of the non-systematic error can be reduced by averaging samples in time or space, thus effectively creating a new dataset that trades accuracy against spatial or temporal resolution.

- Level-4 data products: A level-4 net carbon ecosystem exchange dataset can be created from the combination of a level-3 soil moisture dataset and several ancillary products, namely land surface temperature, vegetation state, and landcover status.

- Multispectral measurements: infrared and microwave snow and ice cover datasets can be combined to produce new, multispectral, more accurate datasets.

- Sharing a common dish: The L-band radar and radiometer can share a common antenna thus effectively reducing the total mass of the system.

- Learning Curve: If several identical satellites are developed and fabricated, the marginal cost of the second and subsequent units is lower than the cost of the first unit due to learning.

4.7. Results

The set of 3,636 architectures was evaluated using VASSAR. The benefit portion is based on the set of requirements described in the Appendix. The cost portion is based on a cost model described in the Appendix. The point science scores and lifecycle cost estimates for all these architectures are shown on Fig. 9. These numbers are obtained from defuzzyfying the fuzzy values. Non-dominated architectures are highlighted in red.

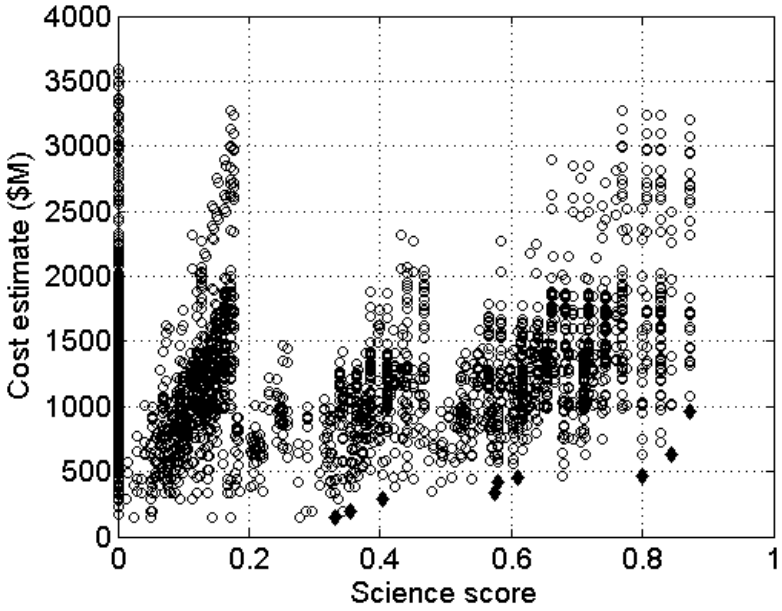


Fig. 9: Science score vs cost estimate for the 3,636 architectures. Diamonds represent non-dominated architectures.

4.7.1. Tradespace analysis

The main goal of any system architecting tool is to gain insight into the "shape" of the tradespace, i.e., what the main trades are, whether there are families of architectures, and so forth. We observe several features just by looking at the tradespace in Fig. 9:

- A very large number of architectures get a science score of 0. This happens when an architecture does not meet one or more requirements that are defined as critical to provide value, or when the instruments are put in environments where they cannot operate. An example of the latter is when a side-looking instrument is put at 400km, resulting in a too large image distortion.

- No architecture gets a perfect science score of 1. This is due to the existence of unresolvable conflicting requirements. In this particular case, most of the observations require a SSO in order to get rid of diurnal variations in radiance. However, a small subset of requirements that concern oceanography or cryospheric measurements are ideally taken in true polar, non-SSO orbits, in order to avoid tidal aliasing (oceanography), or to obtain a better coverage of the polar regions (cryosphere). Since it is impossible to be in a polar and SSO orbit simultaneously, some value is going to be lost no matter what decision is made.

- We observe clusters of architectures that achieve the same science score at different costs. This is a typical behavior in architectural tradespaces, that has its origins in the non-linear mapping between capabilities and satisfaction, namely in the quantization of satisfaction levels. In other words, slightly different performances may be perceived as equivalent in terms of satisfaction by stakeholders.

4.7.2. Use of explanation facility

The explanation facility provides support for more advanced analysis of the tradespace. Examples of the features of the explanation facility are listed below:

- Text-based support: The explanation facility provides detailed explanations of the scores of an architecture in text format as required by the user. An example of such information is provided in Table 2.

Table 2: Example of text-based explanations for science score

Architecture #3 achieves a score of 0.8730 because:

Subobj CLI2-2 (meas "3.4.1 Ocean surface wind speed") gets a score of 0.5 (loss of 0.010 value) because:

Attribute orbit-inclination gets a score of "Half" because of SSO orbit does not provide adequate tidal
sampling (polar orbit required)

Subobj ECO2-1 gets a score of 0 because:

No measurement of parameter "2.3.3 Carbon net ecosystem exchange NEE" is found (requires multispectral measurements)

Subobj WAT3-1 (meas "4.2.4 snow cover") gets a score of 0.415 (loss of 0.013 value) because:

Attribute Accuracy gets a score of "Most" because of Insufficient accuracy (Missing multispectral combination
of sensors)

Attribute orbit-inclination gets a score of "Half" because SSO orbit does not provide adequate coverage of
polar regions (polar orbit required)

Subobj WAT4-1 (meas "4.3.2 Sea ice cover") gets a score of 0.2075 (loss of 0.018 value) because:

Attribute Accuracy gets a score of Some because of Insufficient accuracy (Missing multispectral combination
of sensors)

Attribute orbit-inclination gets a score of "Half" because of SSO orbit does not provide adequate coverage of
polar regions (polar orbit required)

Subobj WEA1-1 (meas "2.3.2 soil moisture") gets a score of 0.83 (loss of 0.020 value) because:

Attribute Horizontal-Spatial-Resolution gets a score of "Half" because of insufficient HSR to meet future
NWP grid size (4km required, [4,12]km achieved)

- Basic graphic-based support: The explanation facility can provide information about one or more architectures on the tradespace just by clicking on the corresponding points on a chart. For example, we can obtain the details of the non-dominated architectures in Fig. 9 by clicking on them, as shown in Table 3. Note that all non-dominated architectures have 800km dawn-dusk SSO, and they are monolithic architectures (all instruments are put onto a single spacecraft). These are thus dominating features. The exact payload composition and the number of satellites in the constellation (1 or 2) vary across the non-dominated set. Note the absence of the IR instrument on non-dominated architectures. This is due to the fact that it has a conflicting orbit requirement with the rest of instruments, which results in an unfavorable science-cost trade. In other words, adding the IR instrument to the suite would require flying the instrument in an AM orbit instead of a dawn-dusk orbit, which would negatively impact both the science output of the other instruments and the cost of the platform.

Table 3: Details of non-dominated architectures from Fig. 7.

Arch#	Payload	Instrument allocation	Orbit altitude	Orbit type	#sats per plane
669	LRADIO XRADIO PSAR	[1;1;1]	800	SSO-DD	2
586	LRADIO PSAR	[1;1]	800	SSO-DD	2
811	LRADIO PSAR	[1;1]	800	SSO-DD	1
2592	LRADIO PSAR	[1]	800	SSO-DD	2
5	PSAR	[1]	800	SSO-DD	1
3605	LSAR LRADIO	[1]	800	SSO-DD	1
3627	LSAR	[1]	800	SSO-DD	1
696	LRADIO	[1]	800	SSO-DD	2
810	LRADIO	[1]	800	SSO-DD	1

The tool can also highlight the regions of the tradespace that correspond to a particular combination of architectural decisions. For example, Fig. 10 highlights all architectures in which the number of satellites per plane is 1. It is easy to see on this chart that it is impossible to achieve the maximum achievable science score with only one satellite per plane, due to unsatisfied temporal resolution requirements.

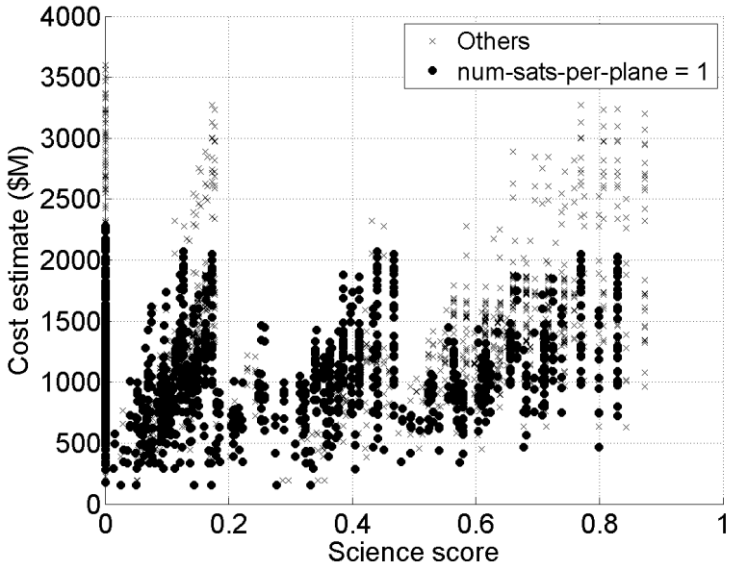


Fig. 10: Architectures with one satellite per plane highlighted

- Advanced graphic-based support: The explanation facility can also provide more advanced support, such as automatic detection of common architectural features in a particular region of the tradespace. For example, if we ask the explanation facility to study the region in which science is in the [0.01; 0.2] interval, the tool compares the attributes of the architectures in this region and detects that most of the architectures have polar, non-SSO orbits, as shown in Fig. 11.

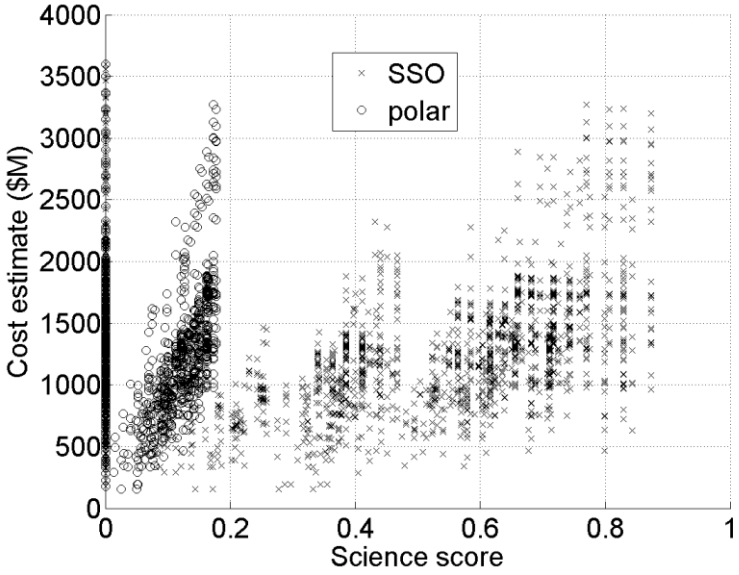


Fig. 11: Effect of orbit inclination on science and cost

4.7.3. Fuzzy results

We emphasized earlier the large uncertainty in the system architecting process, and the importance of being able to deal with fuzzy numbers in the rule-based system. It is important to note that conceptually, we are using fuzzy numbers for two different purposes: capturing uncertainty and capturing fuzziness or ambiguity. Uncertainty refers to statistical uncertainty or randomness, whereas fuzziness refers to non-statistical uncertainty or vagueness. Treating statistical uncertainty with interval analysis provides less information than treating it with probability distribution functions. Put it simply, we only get the boundaries of the probability density function, without any information about its shape. In this example, we are using fuzzy numbers to represent statistical uncertainty in cost, and non-statistical uncertainty in science. Fig. 12 shows the magnitude of the uncertainty for the architectures on the Pareto frontier of Fig. 9. The sources of uncertainty for cost are mostly the standard errors from the cost estimating relationships used in the cost estimation model. The sources of uncertainty for science in this example are the use of fuzzy scores to assess requirement satisfaction (each requirement is satisfied at one of five fuzzy levels as explained in the section describing fuzzy numbers).

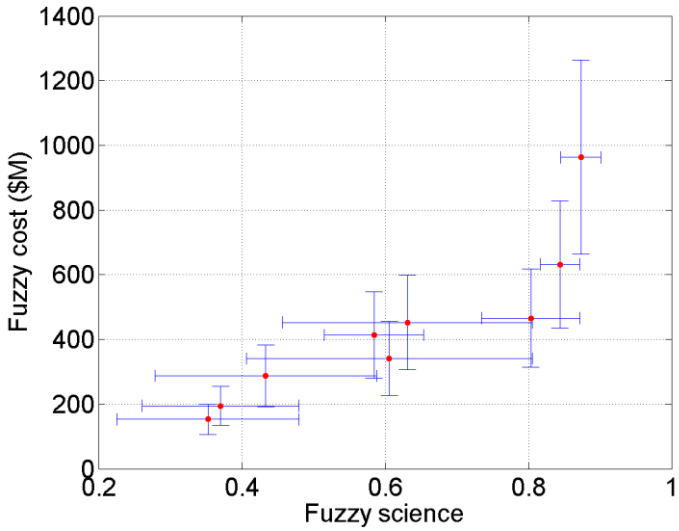


Fig. 12: Fuzzy Science vs fuzzy lifecycle cost for nondominated architectures (uniform weights)

It is important to note that different architectures have different levels of uncertainty. Uncertainty in cost is similar in relative terms (not in absolute terms) across the tradespace because the standard errors of the cost estimating relationships are similar in magnitude (Apgar, 2011). However, uncertainty in science is not homogeneous because a fully satisfied requirement is

encoded as a fuzzy number with mean one and zero width, and a critical requirement that is not satisfied is encoded as a fuzzy number with mean zero and zero width. Thus, for the same score, different uncertainty levels are possible: if the score comes from satisfying a few requirements fully and completely missing the rest, uncertainty will be very low; conversely, if the score comes from satisfying all or most of the requirements at an intermediate level (e.g., "Most" or "Some"), the uncertainty will be much larger.

4.7.4. Sensitivity Analysis

There are several ways of conducting a sensitivity analysis in VASSAR. The most straightforward is simply rerunning the tool with different sets of parameters. In this section we provide two examples of this type of sensitivity analysis: one concerning the capabilities of PSAR to measure soil moisture, and the other one concerning the relative importance of the stakeholder panels.

A major source of uncertainty in this piece of analysis is the ability of the PSAR to provide useful measurements of soil moisture, which has not yet been proven. The appeal of PSAR measurements of soil moisture is that of increased soil and vegetation penetration due to the lower frequency. However, most of the signal at this frequency comes from soil roughness, which makes the soil moisture retrieval challenging (Chalmers University of Technology, 2004). Hence the question of whether this instrument will be able of producing useful measurements of soil moisture is a legitimate one, and it may be interesting to run a pessimistic scenario where the instrument does not have this capability. The details of the non-dominated architectures under this scenario are provided in Table 4.

Table 4: Details of non-dominated architectures when P-band SAR cannot measure soil moisture (uniform weights).

Arch#	Payload	Instrument allocation	Orbit altitude	Orbit type	#sats per plane
1831	LSAR LRADIO XRADIO PSAR	[1;1;2;2]	800	SSO-DD	2
1577	LSAR LRADIO PSAR	[1;1;2]	800	SSO-DD	2
2948	LSAR LRADIO XRADIO PSAR	[1;1;1;2]	800	SSO-DD	1
2631	LSAR LRADIO PSAR	[1;1;2]	800	SSO-DD	1
2604	LSAR LRADIO PSAR	[1;2;2]	800	SSO-DD	1
2592	LSAR LRADIO	[1;1]	800	SSO-DD	2
3605	LSAR LRADIO	[1;1]	800	SSO-DD	1
3627	LSAR	[1]	800	SSO-DD	1
696	LRADIO	[1]	800	SSO-DD	2
810	LRADIO	[1]	800	SSO-DD	1

We note several changes with respect to Table 3. First, the LSAR instrument appears much more often because its combination with LRADIO is the only one that can provide soil moisture measurements that satisfy the needs of the weather community in terms of both accuracy and spatial resolution. The PSAR instrument still appears in the high-cost region of the Pareto frontier, because it is the only instrument with the high penetration capability. However, it disappears of the

lower cost of the Pareto frontier because the extra science does not compensate the cost of developing it for the given stakeholder preferences. It is also noticeable that the high cost non-dominated architectures have now more than one satellite because architectures flying the two SAR on the same platform are dominated (they are too costly).

The second piece of sensitivity analysis models a situation in which the ecosystems panel has become three times as important as the other panels. The details about the non-dominated architectures in this case are shown in Table 5. This change in stakeholder preferences brings forth a major change in the architectural tradespace: the best architectures in the high-science region of the tradespace now include the IR instrument, contrary to what happened for uniform weights. In order to get the maximum science output out of the IR instrument it is necessary to fly it in an AM orbit, which impacts both the science output of other instruments and the cost of the spacecraft.

Table 5: Details of non-dominated architectures when ecosystems panel is 3 times as important as the others

Arch#	Payload	Instrument allocation	Orbit altitude	Orbit type	#sats per plane
540	LRADIO XRADIO IR PSAR	[1;2;2;1]	800	SSO-AM	2
669	LRADIO XRADIO PSAR	[1;1;1]	800	SSO-DD	2
586	LRADIO PSAR	[1;1]	800	SSO-DD	2
811	LRADIO PSAR	[1;1]	800	SSO-DD	1
5	PSAR	[1]	800	SSO-DD	1
3605	LSAR LRADIO	[1;1]	800	SSO-DD	1
3627	LSAR	[1]	800	SSO-DD	1
696	LRADIO	[1]	800	SSO-DD	2

More advanced variants of this option allow computing the threshold value for a parameter that makes the non-dominated set change. For example, the tool determined in this case that when the relative weight of the ecosystems panel w_{ECO} remains below 2.59, the best architectures remain similar to those presented in Table 3, whereas for $w_{ECO} \geq 2.59$, the best architectures switch to those presented in Table 5. Some decision makers find more value in results that provide switching points in a certain trade, rather than point scenario analysis.

A second way of conducting a sensitivity analysis with this tool is by encoding any parameter as a fuzzy value. For instance, the mass or the accuracy of an instrument can also be encoded as a fuzzy number with a certain range of values, and these uncertainties will be propagated to the cost and science metrics. An example of the propagation of uncertainty in mass to cost is shown in Fig. 13. The triangles represent the triangular membership functions for mass and cost. Note that the altitudes of these triangles are notional and do not correspond to the values in the axis. The red line shows the correspondence between scalar mass and scalar cost through a parametric relationship embedded in the tool.

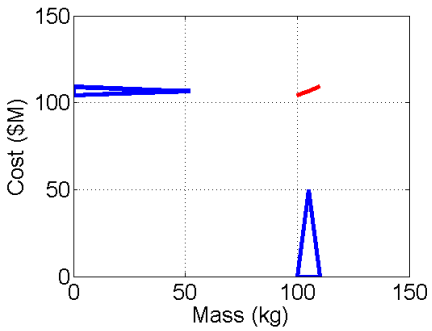


Fig. 13: Propagation of uncertainty in payload mass to payload cost

Finally, a third way to conduct sensitivity analysis with VASSAR is to perform a local search around a particular region or architecture and look at how value delivery to stakeholders change when each variable is changed. Note that gradients or pseudo-gradients cannot generally be defined in this local search process as some of the variables are categorical. Instead, rules are created that automatically enumerate all the architectures that differ from the reference architecture in just one architectural aspect. For example, an instrument is added to or removed from the payload, or the orbit type is changed from SSO-DD to SSO-AM. We used VASSAR to look around the architecture #669 from Table 3. Eleven architectures were thus automatically enumerated and evaluated. They are shown in Fig. 14. This kind of results can be useful to determine the optimal evolution of an architecture, such as in determining an optimal descoping option in the event of a downward budget.

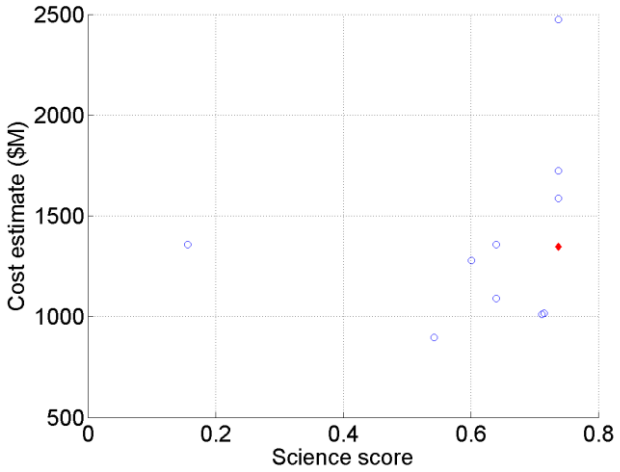


Fig. 14: Local search around architecture #669 from Table 8.

5 Conclusion

This paper has presented a design evaluation methodology that incorporates a rule-based expert system for increased scalability and traceability of the value chain. Such methodology can be used in the context of automatic design or architecture tradespace exploration. After illustrating the methodology with the simple example of a laptop, the tool was demonstrated on a complex example, namely that of an Earth Observing Satellite System.

The rest of this section is divided in two parts. First, the advantages and disadvantages of VASSAR with respect to the state-of-the-art of system architecting tools is discussed. Second, the next steps in this research project are outlined.

5.1. Advantages and disadvantages of VASSAR

The following are advantages (+) and disadvantages (-) of the VASSAR methodology with respect to other frameworks:

- (+) The use of knowledge-based systems decouples the domain-specific knowledge from the domain-independent knowledge, which translates into increased scalability and

reusability, and facilitates task allocation between system architects and software engineers.

- (+) VASSAR has some degree of commonality with the current trend of having databases of lessons learned: it serves as a repository of knowledge to conserve expertise, and it also uses the lessons learned in the form of logical rules to evaluate system architectures.
- (+) VASSAR forces you to articulate the value delivery loop. All requirements are directly traceable to stakeholder needs. All capabilities are directly traceable to architectural decisions.
- (+) Communication between different teams (e.g., science and engineering) is facilitated through use of logical rules that are easy to understand by people from all backgrounds. Indeed, many cognitive psychologists believe that the construct pattern => action is part of how the human mind works (Purves, 2010).
- (+) Traceability of the value delivery loop facilitates more optimal task allocation between man (the system architect) and machine (the computational tool). The computational tool supports the system architect by evaluating a large number of architectures and through the use of the text-based and graphics-based explanation facility. The system architect feeds the tool with knowledge, guides the tradespace exploration process, and makes all non-objective decisions.
- (+) VASSAR uses a functional programming language, which facilitates the design of recursive algorithms, which are at the core of modeling emergence (Poundstone, 1985), which in turn is the origin of value.
- (-) As noted by Minsky amongst others, logical rules are not suited to express all types of expert knowledge (Minsky, 1975).
- (-) Traceability has a computational price, which is only worth it if the knowledge base is large and likely to evolve. If the value chain is simple, or if its traceability is not required, existing tools are more efficient.
- (-) RBES may be slow when the number of rules is very large. As an example, the rule-based system presented in Sect. 4 has about 1,000 rules and takes on the order of a second to evaluate one architecture using a 64-bit quad-core desktop with 12GB of RAM.
- (-) There is an up-front cost to develop the rule-based system and import the expert knowledge.
- (-) Verification of the knowledge base (e.g., completeness, consistency) is challenging and requires extensive testing and/or the development of ad-hoc software.

5.2. Next steps

This paper focused on the design evaluation process. Knowledge-based systems can be used in other aspects of the architecting process. A similar framework has been created that allows automatic enumeration of several canonical types of architectures using rule-based systems (Selva, 2012) and exploration of the resulting architecture spaces. This framework is continuously being improved with more efficient space exploration algorithms for certain classes of architectures.

The extension of VASSAR's explanation facility with more advanced features requires the incorporation of a machine learning layer on top of the rule-based system. This type of hybrid artificial intelligence tool is seen as a potentially fruitful area of research.

The current VASSAR implementation is restrictive in terms of representation of designs, as they need to be represented as lists of pairs *(decision value)*. A more advanced version of the tool could allow importing designs expressed in more powerful representation tools such as SysML or the Object Process Methodology OPM (Dori, 2002).

The version of the framework presented in the paper uses facts (i.e., pairs of slots and values) as the main data structure. While this is sufficient for most applications, the framework would clearly benefit from using a richer set of data structures. We are currently working on a version of the framework that uses Object Oriented Programming to achieve this goal. In particular Java classes are used to define elements, capabilities, decisions, and so forth. The choice of Java was driven by the fact that our language of choice for the rule-based system, Jess, allows seamless interaction with Java objects. However, given that industry has mostly adopted MBSE tools such as SysML for other systems engineering purposes, it makes sense to use these models as data structures instead, and we have given some thought to this possibility. The conclusion of our preliminary

analysis is that while SysML greatly improves some aspects of the framework, it is insufficient in its current implementation to meet the needs of VASSAR. The next few paragraphs discuss why.

Using SysML models as data structures would greatly facilitate the role of manifest, attribute inheritance, and aggregation rules. The hierarchy of elements (e.g., the system contains subsystems, which contain components) is implicit in a SysML model, in particular in a block definition diagram. A single rule that asserts the existence of the subsystems of a system given the existence of the system would be enough.

However, SysML does not explicitly facilitate the definition of models for the system's capabilities (e.g., measurements in the case of the Earth Observing System). Thus, capabilities would need to be represented as standard objects either in a block definition diagram or requirement diagram in a pure SysML framework. Alternatively, they can be kept as facts or other data structures directly in the rule-based system in a hybrid SysML-KBS framework. In either case, capability rules would link the different subsystem objects to their capability objects.

Parametric models describing the calculation of system and subsystem attributes can replace the rules that currently do these computations. However, performance rules computing the attributes of the capabilities objects cannot be replaced unless parametric models are created for native SysML capability objects.

Similarly, the hierarchy of requirements can be defined by a requirement diagram, thus defining the relationships in the aggregation rules. However, to the best of our knowledge, the requirement diagram misses critical aspects of our framework. First, the testcase method provided to check requirement satisfaction has a too rigid interface not accepting partial satisfaction levels. As a conclusion, it does not seem to allow defining requirement satisfaction functions that are arithmetic and/or logical functions of children requirement satisfaction. Third, it also does not allow the definition of stakeholder objects (with their own satisfaction) that own these requirements. And last, but not least, it does not allow to keep the traceability of how the requirement was satisfied.

Finally, none of the diagrams in SysML seem to support the definition of emergence rules that generate new capability objects from combinations of capability objects. While one could argue that this goal could be achieved by using automatic programming techniques that generate SysML diagrams in real time, we believe that the rule-based implementation is much simpler and more elegant.

For all these reasons, we argue that the core of the VASSAR framework would need to remain in the knowledge-based system as opposed to SysML, unless changes are made to the SysML specification.

Finally, we plan to explore multi-agent architectures for the next generation of VASSAR, in order to improve flexibility and allow for the independent development of heterogeneous agents.

Appendix 1: Hierarchy of stakeholder needs for Earth observation example
The information used in the case study for the stakeholders and their hierarchy of requirements are shown below in Table 6 and Table 7.

Table 6. Stakeholders and weights

Panel	Id	Description	Weight
Weather	WEA	Weather	20%
Climate	CLI	Climate	20%
Ecosystems	ECO	Land and Ecosystems	20%
Water	WAT	Water	20%
Applications	HEA	Human health	20%
			100%

Table 7. Stakeholder objectives and weights

Objective	Description	Value
Weather panel		
WEA1	Initialization of NWP models	60%
WEA2	River forecast streamflow models	20%
WEA3	River forecast flash flood models	20%
Climate panel		
CLI1	Boundary conditions for climate models	80%
CLI2	Ocean thermohaline circulation	20%
Ecosystems panel		
ECO1	Net carbon flux in boreal landscapes	75%
ECO2	Carbon net ecosystem exchange	25%
Water panel		
WAT1	Estimation of runoff-EVT	67%
WAT2	Estimation of precipitation	11%
WAT3	Snow and cold land processes	11%
WAT4	Sea Ice cover	11%
Applications panel		
HEA1	Heat Stress and Drought	20%
HEA2	Agriculture productivity	20%
HEA3	Flood monitoring	20%
HEA4	Wild fires prediction	20%
HEA5	Spread of infectious diseases	20%

Appendix 2: Characteristics and capabilities of instruments for Earth observation example

The characteristics of the instruments are provided in the Table 8 and Table 9 .

Table 8: Instrument characteristics

	LRADIO	LRADAR	XRADIO	IR	PRADIO
Mass (kg)	202	236	257	199	390
Avg. power (W)	67	440	340	134	430
Avg. data rate (Mbps)	20	6	0.3	6.5	80

Table 9: Instrument capabilities

Parameter	LRADIO	LRADAR	XRADIO	IR	PRADIO
Soil Moisture	High accuracy, low spatial resolution	Low accuracy, high spatial resolution	Medium accuracy, low spatial resolution	Lowest accuracy, high spatial resolution	Highest accuracy, low spatial resolution
Freeze-Thaw		High accuracy, high spatial resolution			
Snow cover	Medium accuracy, low spatial resolution	Low accuracy, high spatial resolution	High accuracy, low spatial resolution	Low accuracy, high spatial resolution	Medium accuracy, low spatial resolution
Sea ice cover	Medium accuracy, low spatial resolution	Low accuracy, high spatial resolution	High accuracy, low spatial resolution	Low accuracy, high spatial resolution	Medium accuracy, low spatial resolution
Sea surface wind	Low accuracy, low spatial resolution	Low accuracy, high spatial resolution	Highest accuracy, low spatial resolution		Low accuracy, low spatial resolution
Precipitation			Highest		

rate			accuracy, low spatial resolution		
Ocean salinity	High accuracy, low spatial resolution	Medium accuracy, high spatial resolution			Low accuracy, low spatial resolution

Appendix 3: Cost model

The cost model used in the case study is a rule-based cost model largely based on Larson and Wertz's Space Mission Analysis and Design. The first level decomposition of lifecycle cost is given in Fig. 15.

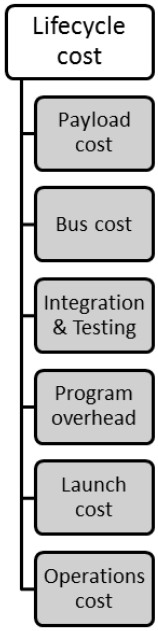


Fig. 15: Lifecycle cost decomposition

Payload cost is based on the NASA Instrument Cost model (Habib-Agahi, Ball, & Fox, 2009). Bus cost is based on the parametrics provided in (Apgar, 2011). Since these parametrics are based on the spacecraft mass budget, a spacecraft design module that estimates the mass and power budgets of each spacecraft precedes the cost estimation module.

The spacecraft design module is iterative because of the couplings between different subsystems. For example, the mass of the spacecraft affects the design of the ADCS through the size of the reaction wheels and the amount of propellant amongst others, and these feed back into the computation of the spacecraft mass. In practice, three iterations are sufficient to make the design process converge to a precision of less than a kg. An overview of the spacecraft design module is provided in Fig. 16.

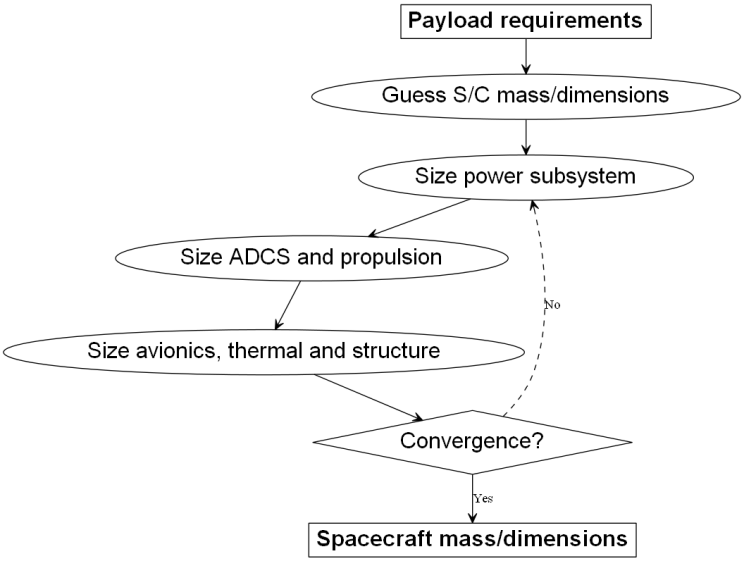


Fig. 16: Spacecraft design algorithm

References

Antonsson, E. K., & Cagan, J. (2001). *Formal Engineering Design Synthesis*. Cambridge: Cambridge University Press. doi:10.1017/CBO9780511529627

Apgar, H. (2011). Cost Estimating. In *Space Mission Engineering: The new SMAD*. Hawthorne, CA: Microcosm.

Armacost, R. L., Componation, P. J., Mullens, M. A., & Swart, W. S. (1994). An AHP framework for prioritizing customer requirements in QFD: an industrialized housing application. *IIE Transactions*, *26*(4), 72–79.

Avigad, G., & Moshaiov, A. (2009). Interactive evolutionary multiobjective search and optimization of set-based concepts. *IEEE Transactions on Systems, Man, and Cybernetics. Part B, Cybernetics*, *39*(4), 1013–27. doi:10.1109/TSMCB.2008.2011565

Bellman, R. E., & Zadeh, L. A. (1970). Decision-making in a fuzzy environment. *Management Science*, *17*(4), B–141.

Bonczek, R. H., Holsapple, C. W., & Whinston, A. B. (1981). A Generalized Decision Support System Using Predicate Calculus and Network Data Base Management. *Operations Research*, *29*(2), 263–281. doi:10.2307/170020

Buchanan, B. G., & Shortliffe, E. H. (1984). *Rule-based Expert Systems: the MYCIN experiments of the Stanford Heuristic Programming Project*. *Language*. Addison-Wesley.

Cameron, B. (2008). Value flow mapping: Using networks to inform stakeholder analysis. *Acta Astronautica*, *62*(4-5), 324–333. doi:10.1016/j.actaastro.2007.10.001

Campbell, M., Cagan, J., & Kotovsky, K. (1999). A-design: An agent-based approach to conceptual design in a dynamic environment. *Research in Engineering Design*, 172–192.

Carlson-Skalak, S., White, M., & Teng, Y. (1998). Using an evolutionary algorithm for catalog design. *Research in Engineering Design*, 63–83.

Chalmers University of Technology. (2004). *Use of P-band SAR for forest biomass and soil moisture retrieval*. Retrieved from http://esamultimedia.esa.int/docs/gsp/completed/C16115ExS.pdf

Chandrasekaran, B. (1989). A framework for design problem-solving. *Research in Engineering Design*, *1*, 75–86.

Charnes, A., & Cooper, W. W. (1957). Management models and industrial applications of linear programming. *Management Science*, *4*(1), 38–91.

Clancey, W. J. (1987). Knowledge-Based Tutoring: The GUIDON program. In *MIT Press Series in Artificial Intelligence*. Cambridge, MA: The MIT Press.

Corkill, D. D. (2003). Blackboard and Multi-Agent System & the Future. In *Proceedings of the International Lisp Conference*. New York, New York. Retrieved from http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Blackboard+and+Multi-Agent+Systems+&+the+Future#5

Crawley, E., De Weck, O., Eppinger, S., Magee, C., Moses, J., Seering, W., … Whitney, D. (2004). *The influence of architecture in engineering systems - Engineering Systems Monograph*. *Architecture*.

Das, I., & Dennis, J. E. (1998). Normal-Boundary Intersection: A New Method for Generating the Pareto Surface in Nonlinear Multicriteria Optimization Problems. *SIAM Journal on Optimization*, *8*(3). doi:10.1137/S1052623496307510

De Condorcet, M. (1785). Essai sur l'application de l'analyse à la probabilité des décisions rendues à la probabilité des voix. *De l'Imprimerie Royale, Paris*.

De Weck, O. L., & Kim, I. Y. (2004). Adaptive weighted sum method for bi-objective optimization. In *Proceedings of the 45th AIAA/ASME/ASCE/* (pp. 1–13).

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182–197. doi:10.1109/4235.996017

Dincbas, M. (1980). A knowledge-based expert system for automatic analysis and synthesis in CAD. In *IFIP Congress* (pp. 705–710).

Dobias, A. P. (1990). Designing a mouse trap using the analytic hierarchy process and expert choice. *European Journal of Operational Research*, *48*(1), 57–65.

Donaldson, K. M., Ishii, K., & Sheppard, S. D. (2006). Customer Value Chain Analysis. *Research in Engineering Design*, *16*(4), 174–183. doi:10.1007/s00163-006-0012-8

Dori, D. (2002). *Object-Process Methodology: A holistic paradigm* (pp. 1–453). Berlin, Heidelberg: Springer.

Duda, R. 0., Gaschnig, J. G., & Hart, P. E. (1979). Model Design in the PROSPECTOR Consultant System for Mineral Exploration. In D. Michie (Ed.), *Expert Systems in the Microelectronic Age* (pp. 153–167). Edinburgh, Scotland: Edinburgh University Press.

Durkin, J. (1990). Application of Expert Systems in the Sciences. *Ohio Journal of Sciences*, *90*(5), 171–179.

Engelmore, R., & Morgan, T. (Eds.). (1988). *Blackboard systems* (pp. 1–620). Addison Wesley Publishing Company.

Entekhabi, D. (2010). The Soil Moisture Active Passive (SMAP) Mission. In *Proceedings of the IEEE* (Vol. 98, pp. 704–716). doi:10.1109/JPROC.2010.2043918

Erman, L., & Hayes-Roth, F. (1980). The Hearsay-II speech-understanding system: Integrating knowledge to resolve uncertainty. *ACM Computing Surveys*, *12*(2), 213–253. Retrieved from http://dl.acm.org/citation.cfm?id=356816

Feigenbaum, E. A., Buchanan, B. G., & Lederberg, J. (1971). On generality and problem solving: a case study using the DENDRAL program. In B. Meltzer & D. Michie (Eds.), *Machine Intelligence 6* (pp. 165–190). Edinburgh, Scotland.

Fenves, S. ., & Garrett, J. . (1986). Knowledge based standards processing. *Artificial Intelligence in Engineering*, *1*(1), 3–14. doi:10.1016/0954-1810(86)90029-4

Fishburn, P. C. (1974). Lexicographic orders, utilities and decision rules: A survey. *Management Science*, 1442–1471.

Fortin, J., Dubois, D., & Fargier, H. (2008). Gradual Numbers and Their Application to Fuzzy Interval Analysis. *IEEE Transactions on Fuzzy Systems*, *16*(2), 388–402. doi:10.1109/TFUZZ.2006.890680

Gasster, S., & Flaming, G. G. M. (1998). Overview of the conical microwave imager/sounder development for the NPOESS program. In *Geoscience and Remote Sensing …* (Vol. 1, pp. 268–270). Ieee. doi:10.1109/IGARSS.1998.702874

Geoffrion, A. M., Dyer, J. S., & Feinberg, A. (1972). An interactive approach for multi-criterion optimization, with an application to the operation of an academic department. *Management Science*, 357–368.

Giarratano, J. C., & Riley, G. D. (2004). *Expert Systems: Principles and Programming* (Fourth Edi.). Course Technology. Retrieved from http://www.amazon.com/Expert-Systems-Principles-Programming-Fourth/dp/0534384471

Gologlu, C., & Mizrak, C. (2011). An integrated fuzzy logic approach to customer-oriented product design. *Journal of Engineering Design*, *22*(2), 113–127. doi:10.1080/09544820903032519

Gutknecht, O., & Ferber, J. (2001). The MadKit Agent Platform Architecture. In T. Wagner & O. Rana (Eds.), *Infrastructure for Agents, Multi-Agent Systems, and Scalable Multi-Agent Systems SE - 5* (Vol. 1887, pp. 48–55). Springer Berlin Heidelberg. doi:10.1007/3-540-47772-1_5

Habib-Agahi, H., Ball, G., & Fox, G. (2009). NICM Schedule & Cost Rules of Thumb. In *AIAA Space Conference 2009* (pp. 6512–6512). Pasadena, CA: AIAA.

Hart, P., Duda, R., & Einaudi, M. (1978). PROSPECTOR—A computer-based consultation system for mineral exploration. *Mathematical Geology*, (November 1977). Retrieved from http://www.springerlink.com/index/3520V0M3W1864773.pdf

Haskins, C. (2006). *INCOSE Systems engineering handbook - A guide for system life cycle processes and activities* (No. INCOSE-TP-2003-002-03). International Council on Systems Engineering.

Hauser, J., & Clausing, D. (1988). The house of quality. *Harvard Business Review*, (May-June 1998).

Heliere, F., Lin, C. C., Fois, F., Davidson, M., Thompson, A., & Bensi, P. (2009). BIOMASS: A P-band SAR earth explorer core mission candidate. In *Proceedings of the 2009 IEEE Radar Conference*. Ieee. doi:10.1109/RADAR.2009.4977088

Ignizio, J. (1983). Generalized goal programming - An overview. *Computers & Operations Research*, *I*(4).

Keeney, R. L., & Raiffa, H. (1976). *Decisions with multiple objectives: preferences and value trade-offs* (p. 592). New York: Wiley.

Koo, B. H. Y., Simmons, W. L., & Crawley, E. F. (2009). Algebra of Systems: A Metalanguage for Model Synthesis and Evaluation. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, *39*(3), 501–513. doi:10.1109/TSMCA.2009.2014546

Kurtoglu, T., & Campbell, M. I. (2009). An evaluation scheme for assessing the worth of automatically generated design alternatives. *Research in Engineering Design*, *20*(1), 59–76. doi:10.1007/s00163-008-0062-1

Lai, Y. J., Liu, T. Y., & Hwang, C. L. (1994). Topsis for MODM. *European Journal of Operational Research*, *76*(3), 486–500.

Liao, S. (2005). Expert system methodologies and applications—a decade review from 1995 to 2004. *Expert Systems with Applications*, *28*(1), 93–103. doi:10.1016/j.eswa.2004.08.003

Lighthill, J. (1973). Artificial Intelligence: A General Survey. In B. S. R. Council (Ed.), *Artificial Intelligence: a Paper Symposium*.

Lindsay, R., Buchanan, B. G., & Feigenbaum, E. A. (1993). DENDRAL: A Case Study of the First Expert System for Scientific Hypothesis Formation. *Artificial Intelligence*, *61*(2), 209–261. doi:10.1016/0004-3702(93)90068-M

Malen, D. E., & Hancock, W. M. (1995). Engineering for the customer: Combining preference and physical systems models: Part I-theory. *Journal of Engineering Design*, *6*(4), 315–328.

Mattson, C. a., & Messac, A. (2003). Concept Selection Using s-Pareto Frontiers. *AIAA Journal*, *41*(6), 1190–1198. doi:10.2514/2.2063

Mauchand, M., Siadat, A., Bernard, A., & Perry, N. (2008). Proposal for tool-based method of product cost estimation during conceptual design. *Journal of Engineering Design*, *19*(2), 159–172. doi:10.1080/09544820701802857

McDermott, J. (1982). R1: A Rule-Based Configurer of Computer Systems. *Artificial Lntell., 19: 39*, *19*(1), 39–88. doi:10.1016/0004-3702(82)90021-2

Messac, A., & Ismail-Yahaya, A. (2002). Multiobjective robust design using physical programming. *Structural and Multidisciplinary Optimization*, *23*(5), 357–371. doi:10.1007/s00158-002-0196-0

Messac, A., Ismail-Yahaya, A., & Mattson, C. A. (2003). The normalized normal constraint method for generating the Pareto frontier. *Structural and Multidisciplinary Optimization*, *25*(2), 86–98. doi:10.1007/s00158-002-0276-1

Minsky, M. (1975). A framework for representing knowledge. In P. H. Whinston (Ed.), *The Psychology of Computer Vision* (pp. 1–81). New York, NY: McGraw-Hill Book.

Mon, D. L., Cheng, C. H., & Lin, J. C. (1994). Evaluating weapon system using fuzzy analytic hierarchy process based on entropy weight. *Fuzzy Sets and Systems*, *62*(2), 127–134.

Newell, A., & Simon, H. A. (1972). *Human Problem Solving*. Englewood Cliffs, NJ: Prentice Hall.

Nii, H. (1986). The blackboard model of problem solving and the evolution of blackboard architectures. *AI Magazine*, *7*(2), 38–53. Retrieved from http://www.aaai.org/ojs/index.php/aimagazine/article/viewArticle/537

Parasuraman, R., Sheridan, T. B., & Wickens, C. D. (2000). A model for types and levels of human interaction with automation. *IEEE Transactions on Systems, Man, and Cybernetics. Part A, Systems and Humans*, *30*(3), 286–97.

Pareto, V. (1896). *Cours d'economie politique*. Geneva: Librairie Droz.

Park, J., & Han, S. H. (2004). A fuzzy rule-based approach to modeling affective user satisfaction towards office chair design. *International Journal of Industrial Ergonomics*, *34*(1), 31–47.

Poundstone, W. (1985). *The Recursive Universe: Cosmic Complexity and the Limits of Scientific Knowledge* (pp. 1–252). Contemporary Books.

Pugh, S. (1991). *Total design: integrated methods for successful product engineering*. Workingham: Addison Wesley Publishing Company.

Purves, D. (2010). *Brains: How they seem to work* (pp. 1–320). FT Press.

Radovcic, Y., & Remouchamps, A. (2002). BOSS QUATTRO: an open system for parametric design. *Structural and Multidisciplinary Optimization*, *23*(2), 140–152. doi:10.1007/s00158-002-0173-7

Reich, Y. (2010). My method is better! *Research in Engineering Design*, *21*(3), 137–142. doi:10.1007/s00163-010-0092-3

Rohl, P. J., Kolonay, R. M., Irani, R. K., Sobolewski, M., Kao, K., & Bailey, M. W. (2000). A Federated Intelligent Product EnviRonment. In *8th Symposium on Multidisciplinary Analysis and Optimization*.

Ross, A. M., Hastings, D. E., Warmkessel, J. M., & Diller, N. P. (2004). Multi-attribute Tradespace Exploration as Front End for Effective Space System Design. *Journal of Spacecraft and Rockets*, *41*(1), 20–28.

Saaty, T. L. (1977). A scaling method for priorities in hierarchical structures. *Journal of Mathematical Psychology*, *15*(3), 234–281. doi:http://dx.doi.org/10.1016/0022-2496(77)90033-5

Saaty, T. L. (1990). How to make a decision: The analytic hierarchy process. *European Journal of Operational Research*, *48*(1), 9–26. doi:10.1016/0377-2217(90)90057-I

Salomon, S., Dom, C., Avigad, G., Freitas, A., Goldvard, A., & Sch, O. (2014). PSA Based Multi Objective Evolutionary Algorithms. In O. Schuetze, C. A. Coello Coello, A.-A. Tantar, E. Tantar, P. Bouvry, P. Del Moral, & P. Legrand (Eds.), *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation III* (pp. 233–255). Heidelberg: Springer International Publishing. doi:10.1007/978-3-319-01460-9

Schreiber, G., Akkermans, H., Anjewierden, A., de Hoog, R., Shadbolt, N., Van De Velde, W., & Wielinga, B. (2000). *Knowledge Engineering and Management: The CommonKADS Methodology*. Cambridge, MA: MIT Press.

Seher, T. (2009). *Campaign-level Science Traceability for Earth Observation System Architecting* (MS Thesis). Dept. of Aeronautics and Astronautics. Retrieved from http://dspace.mit.edu/handle/1721.1/51639?show=full

Selva, D. (2012). *Rule-based system architecting of Earth observation satellite systems* (PhD dissertation, Massachusetts Institute of Technology). ProQuest/UMI, Ann Arbor.

Selva, D., & Crawley, E. (2013). VASSAR: Value Assessment of System Architectures using Rules. In *Aerospace Conference, 2013 IEEE*. Big Sky: IEEE.

Shah, J. J., & Rogers, M. T. (1993). Assembly modeling as an extension of feature-based design. *Research in Engineering Design*, *5*(3), 218–237.

Smith, P., & Reinertsen, D. (1997). *Developing products in half the time: new rules, new tools* (2nd Editio.). London, UK: Wiley.

Spanoudakis, G., Zisman, A., Pérez-Miñana, E., & Krause, P. (2004). Rule-based generation of requirements traceability relations. *Journal of Systems and Software*, *72*(2), 105–127. doi:10.1016/S0164-1212(03)00242-5

Stewart, T. (1992). A critical survey on the status of multiple criteria decision making theory and practice. *Omega*, *20*(5-6), 569–586. doi:10.1016/0305-0483(92)90003-P

Stiny, G. (1980). Introduction to shape and shape grammars. *Environment and Planning B: Planning and Design*, *7*(3), 343–351. doi:10.1068/b070343

Suh, N. (1998). Axiomatic design theory for systems. *Research in Engineering Design*, *10*(4), 189–209.

Taguchi, G., Elsayed, E., & Hsiang, T. (1989). *Quality engineering in production systems*. New York: McGraw-Hill.

Thurston, D. L. (1991). A formal method for subjective design evaluation with multiple attributes. *Research in Engineering Design*, *3*(2), 105–122. doi:10.1007/BF01581343

Ulrich, K. (1995). The role of product architecture in the manufacturing firm. *Research Policy*, *24*(3), 419–440. doi:10.1016/0048-7333(94)00775-3

Von Neumann, J., & Morgenstern, O. (1944). *Theory of Games and Economic Behavior* (p. 625). Princeton University Press.

Welsch, C., & Swenson, H. (2001). VIIRS (Visible Infrared Imager Radiometer Suite): a next-generation operational environmental sensor for NPOESS. *Proceedings of the 2001 International Geoscience and Remote Sensing Symposium*, *3*, 1020–1022.

Wnuk, K., Regnell, B., & Schrewelius, C. (2009). Architecting and Coordinating Thousands of Requirements – An Industrial Case Study. In M. Glinz & P. Heymans (Eds.), *Requirements Engineering: Foundation for Software Quality SE - 10* (Vol. 5512, pp. 118–123). Springer Berlin Heidelberg. doi:10.1007/978-3-642-02050-6_10

Wolfram, S. (2002). *A new kind of science*. Champaign, IL: Wolfram Media, Inc.

Yager, R. (1988). On ordered weighted averaging aggregation operators in multicriteria decision making. *Systems , Man and Cybernetics, IEEE Transactions on*, (1), 183–190.

Yager, R. R. (1977). Multiple objective decision-making using fuzzy sets. *International Journal of Man-Machine Studies*, *9*(4), 375–382.

Zadeh, L. (1963). Optimality and Non-Scalar-Valued Performance Criteria. *IEEE Transactions on Automatic Control*, *59*.

Zadeh, L. A. (1965). Fuzzy Sets. *Information and Control*, *8*(3), 338–353. doi:10.1016/0165-0114(78)90029-5

Zimmerman, H. J. (1983). Using fuzzy sets in operational research. *European Journal of Operational Research*, *13*(3), 201–216.

Ziv-Av, A., & Reich, Y. (2005). SOS – subjective objective system for generating optimal product concepts. *Design Studies*, *26*(5), 509–533. doi:10.1016/j.destud.2004.12.001