VASSAR: Value Assessment of System Architectures using Rules

Daniel Selva
Aeronautics and Astronautics
Massachusetts Institute of Technology
Cambridge, MA 02139
617-682-6521
dselva@mit.edu

Edward F. Crawley
Aeronautics and Astronautics
Massachusetts Institute of Technology
Cambridge, MA 02139
617-253-7510
crawley@mit.edu

***Abstract*—A key step of the mission development process is the selection of a system architecture, i.e., the layout of the major high-level system design decisions. This step typically involves the identification of a set of candidate architectures and a cost-benefit analysis to compare them. Computational tools have been used in the past to bring rigor and consistency into this process. These tools can automatically generate architectures by enumerating different combinations of decisions and options. They can also evaluate these architectures by applying cost models and simplified performance models.**

Current performance models are purely quantitative tools that are best fit for the evaluation of the technical performance of a mission design. However, assessing the relative merit of a system architecture is a much more holistic task than evaluating the performance of a mission design. Indeed, the merit of a system architecture comes from satisfying a variety of stakeholder needs, some of which are easy to quantify, and some of which are harder to quantify (e.g., elegance, scientific value, political robustness, flexibility). Moreover, assessing the merit of a system architecture at these very early stages of design often requires dealing with a mix of: a) quantitative and semi-qualitative data; b) objective and subjective information. Current computational tools are poorly suited for these purposes.

In this paper, we propose a general methodology that can be used to assess the relative merit of several candidate system architectures under the presence of objective, subjective, quantitative, and qualitative stakeholder needs. The methodology is called VASSAR (Value ASsessment for System Architectures using Rules). The major underlying assumption of the VASSAR methodology is that the merit of a system architecture can be assessed by comparing the capabilities of the architecture with the stakeholder requirements. Hence for example, a candidate architecture that fully satisfies all critical stakeholder requirements is a good architecture. The assessment process is thus fundamentally seen as a pattern matching process where capabilities match requirements, which motivates the use of rule-based expert systems (RBES). This paper describes the VASSAR methodology and shows how it can be applied to a large complex space system, namely an Earth observation satellite system. Companion papers show its applicability to the NASA space communications and navigation program and the joint NOAA-DoD NPOESS program.

TABLE OF CONTENTS

978-1-4673-1813-6/13/$31.00 ©2013 IEEE.
[1] IEEEAC Paper #2255, Version 4, Updated 01/05/2013.

1. INTRODUCTION

System Architecting

System architecting is by definition the process by which a system architecture is defined. Crawley defines a system architecture as an abstract description of the entities of a system and the relationships between those entities [1]. Hence, a system architecture can be seen as the set of high-level early system design decisions. These decisions are important because they will define most of the system's lifecycle properties, and its ability to sustainably provide value to the stakeholders.

Themes in System Architecting

There are several recurring themes in system archictecting that are of relevance to this paper, namely: complexity, emergence, and the value delivery loop.

Complexity and Emergence— Many definitions of system complexity exist that can appear to be quite different in nature (compare for example, [1] and [2]). However, most authors identify two major sources of system complexity:

1) The number of parts of a system and the degree to which these parts are interconnected or coupled.

2) Some degree of uncertainty or unpredictability on the system's behavior, that is sometimes labeled as "non-linear dynamics". Due to this uncertainty, the behavior of a certain system cannot be explained by simple combination of the behavior of its parts. Note that the latter statement usually appears in definitions of the word "system".

Regardless of the definition, architecting complex systems requires a means of dealing with these two aspects, namely representing (and computing with) hierarchies of elements, and modeling emergent behavior. For example, a form decomposition of a system can be represented using a 3-level tree, in which the root tree is the system; the system is divided in subsystems; and each subsystem consists of components. Modeling emergence can be done for example by means of simulation, or simple logical rules: "if I can run, I can swim, and I can cycle, I can participate in a triathlon".

The Value Delivery Loop—One of the key characteristics of system architecting is that it strives to maximize stakeholder **value**, rather than just economic profit or performance. Value is often defined as benefit at cost, where benefit goes beyond traditional performance measurements, and includes the satisfaction of both technical and non-technical stakeholder requirements. Hence, a good architecture is one that is capable of sustainable value delivery to the stakeholders over its lifecycle.

The value delivery loop is a path from the system's elements of function and form to the value delivered to the stakeholders. A simplified view of the loop starts with the system which decomposes into a set of subsystems and components. These components perform functions at certain levels of performance, thus satisfying the needs of different stakeholders. Overall, the value delivery loop goes from the system's form domain, to its function domain, to stakeholder satisfaction.

Role of Computational Tools in System Architecting

Much of the work in the field of systems architecting has been qualitative in nature, describing principles, methods, and tools that can be applied to architect a system. Examples of this are references collecting "principles" of good system architecting ([3], [4]).

Computational tools are currently utilized for system architecting purposes in many organizations. These tools typically focus on **describing** a complex system architecture, i.e., they focus on the first aspect mentioned in the previous section ([5], [6]). Their major role is to help communicate a certain system architecture.

A subset of these tools incorporate a simulation layer on top of the descriptive layer. These tools are executable and allow simulation of the system's behavior in different operational scenarios ([7], [8]). These tools can thus tackle the second aspect identified earlier: modeling emergent behavior.

Relatively little effort has been put on the development of quantitative tools tailored to **optimize** a system architecture, or equivalently, to explore the tradespace of possible system architectures. The terms architectural tradespace exploration and architecture optimization are used as synonyms in this paper, and designate the global effort to identify preferred system architectures, study the relative importance of and the coupling between decisions, and more generally, gain insight about the shape of the architecture trade space (i.e., localize interesting and uninteresting regions, compute sensitivities of figures of merit to decisions, and so forth).

Computational tools for architecture tradespace exploration perform three main functions:

(1) **Enumeration** or synthesis of architectures, based on a mathematical model (usually a graph) containing the main architectural decisions to be made, and their set of allowed options.

(2) **Evaluation** of architectures, based on a mix of objective and subjective figures of merit.

(3) **Down-selection** of architectures, based on a mix of objective and subjective criteria.

There are several likely reasons as to why little has been published on these tools, mostly related to perceived similarities with design optimization tools, increased uncertainty and ambiguity with respect to other design optimization problems, and often increased difficulty to solve the optimization problem.

However, these tools are important because they can support the system architect by bringing rigor and exhaustiveness to the architecting process, where most of the system's lifecycle cost is committed. This paper outlines a methodology to design a key part of these tools: the module evaluating a system architecture. Due to the particularities of system architecting with respect to system design (mixed use of subjective and objective information, mixed use of quantitative and qualitative information, high uncertainty and ambiguity, complex value delivery loop), a specific technology (namely rule-based expert systems) was used. The reasons for this choice are outlined in the following sections.

Value Assessment of System Architectures

The evaluation function of a tradespace exploration tools requires a way of quantifying the value of a system architecture through the use of objective functions that capture the needs and requirements of stakeholders. Objective functions in system architecting are often called **value functions**. Creating a set of value functions can be challenging because some aspects of the value definition may be subjective or hard to quantify, e.g., "the value of a scientific discovery", or "the engagement of the youth". Indeed, evaluating a system architecture requires dealing simultaneously with quantitative and qualitative knowledge.

This section reviews the state-of-the-art of the literature concerning formal methods to estimate the sometimes subjective value of a system architecture. Some may argue that this is an attempt to "quantify the unquantifiable". In reality, the system architect still has to make architectural decisions even in the presence of ambiguity and subjectivity, and therefore we argue that there is value in developing formal methods to help them make these important decisions.

Since the field of system architecture is reduced, related bodies of research were reviewed. In particular, we found that this problem is similar in some aspects to the problems of assessing the worth of a financial endeavor, the value of a new policy or public good, and the technical performance of a system, and therefore we reviewed strategies from economics, policy, and science and engineering.

Value Assessment in Science and Engineering— The most common approach to assess the value of an engineering system (e.g. a satellite mission) is the use of system simulation. In particular, **end-to-end simulation models** simulate everything from the externally delivered function of the system to its supporting functions and its interaction with the surrounding context at a sufficient level of detail that it permits calculation of system performance. This approach is typically used in most science and engineering projects (e.g., [9]) because it is almost always possible to build a model that simulates system behavior, provides cardinally meaningful scores, and is objective.

A different simulation-based approach to value assessment in science is the use of **Observing System Simulation Experiments** or OSSEs. OSSEs are used to quantify the value of a certain dataset, typically in Numerical Weather Prediction (NWP). In OSSEs, a complex forward model simulates the state of the weather system (nature run) and then assesses the value of the dataset by computing the difference in forecasting performance between the whole system with and without the dataset [10]. OSSEs are extensively used in NWP because of their fidelity and objectiveness. However, they are extremely computationally expensive, and they are only applicable to a particular class of systems, namely systems that produce a dataset that is used in a forecasting process.

End-to-end simulation models and OSSEs are objective tools that are not designed to deal with subjective value. Subjective value in science is often assessed by conducting expert surveys. An example of this is NASA's **Science Value Matrices** (SVMs) [11].

Subjective value of engineering products can be assessed using customer input. An instance of tool that falls under this category is the **House of Quality** [12], in which users rate the importance of different customer attributes, while expert engineers rate the coupling between customer attributes and design features. The Analytic Hierarchy Process[13] is another formal methodology to elicit relative preferences. In the presence of more than one metric, multi-attribute utility theory [14] is used to combine multiple utility functions to create a single metric. This combination is possible under a certain set of assumptions.

Value Assessment in Business Management—The canonical tool to assess the value of a project in business management is to compute its **Net Present Value** (NPV). NPV is the sum of all cash flows $b_i - c_i$ time-discounted using a certain discount rate r: $NPV = \sum_i \frac{(b_i - c_i)}{(1+r)^{t_i}}$.

NPV has the advantage of providing a cardinal ranking of architectures, i.e., differences in dollars between the values of two architectures are meaningful. The main disadvantage of this approach is that some systems are less amenable to monetization. For example, it is hard to monetize the value of science, or the inspiration of the youth, two examples used before to illustrate the concept of subjectivity in system architecting. There is also some controversy about the choice of adequate discount rates, and even the adequacy of the concept of discount rates for valuation of certain projects, especially projects funded by taxpayer money [15].

The use of NPV described in the previous paragraph does not include uncertainty. One way of introducing uncertainty is to construct a **decision tree** where the NPVs of each project under a set of plausible scenarios are calculated. Each scenario is then assigned a certain probability of occurrence, and the expected NPV ($E[NPV]$) is computed as the average of the NPVs in each scenario weighted by the probability of occurrence of the scenarios.

Finally, in cases where the product delivered by the system is information, its value can be computed as the $\Delta E[NPV]$ between a case where the information is available, and the reference case where it is not available. Information provides value by updating the probabilities of each scenario, thus leading to an increase in $E[NPV]$. Although the information also has a cost, this cost may be offset by the $\Delta E[NPV]$. This approach is called the **value-of-information** approach (VoI) and it has been successfully applied in the past to climate datasets [16].

Value Assessment of Public Goods—Government organizations often deal with projects whose value cannot be easily monetized, or even whose value is subjective. In this case, the typical source of information for project valuation is expert or stakeholder judgment, i.e., directly surveying the experts or the stakeholders. Expert opinion is elicited through questionnaires and then quantified using scoring rubrics, Likert-based scales, utility theory, or other formal processes such as the contingent valuation method [17], [18].

Decision trees and VoI can also be applied with utility instead of currency. Hence, the metrics for these methods become expected utility $E[u]$ or the $\Delta E[u]$ between the cases with and without the information. For example, Nordhaus applied the utility-based VoI approach to assign a value to a certain climate satellite dataset. The dataset reduces the uncertainty in key climate parameters such as the sensitivity of global temperature to atmospheric CO_2 concentration, thus allowing for a more accurate forecasting of increases in temperature and consequently, a more efficient climate policy design [16].

The advantage of expert judgment based approaches is that they are easy to apply, and they can deal with subjective information. The main disadvantage is that most researchers agree that these methods can only provide an ordinal ranking of architectures, and therefore differences in utility between architectures are not meaningful (only the ordering is). Another problem is that large quantities of expert knowledge may be required in some cases, so that the knowledge elicitation process can actually become the bottleneck of the architecting process.

Classification of valuation strategies—In this subsection we discuss insight gained from the survey of approaches described in the previous subsections. We start by providing a classification of valuation approaches according to two criteria: the units in which benefit is expressed, and the role of the system being assessed.

Strategies to approach this problem can be classified according to their computation of value, and the unit in which value is expressed. We distinguish between three main strategies: a) value is profit (dollars), i.e., benefit minus cost; b) value is performance (physical units) at cost ; c) value is utility (utils) at cost. There is a loose correspondence between these strategies and the fields that inspired them: value is profit in business management; value is performance in science and engineering; value is utility in public policy.

An alternative classification of strategies can be done according to the role of the system being assessed. We distinguish between direct and indirect strategies. In direct strategies, the value of the system is the value directly delivered by the system to the stakeholders. The system takes thus a primary role in the value delivery process. Examples of direct methods are the NPV approach or end-to-end simulation models.

In indirect or delta strategies, the system provides value by being a part of, or a complement to, a larger system. The system takes thus a secondary role in the value delivery process. In this case, the value of the architecture is computed as the marginal (or delta) value that the larger or primary system provides to the stakeholders when the system being assessed is implemented. Examples of indirect methods are the VoI approach and OSSEs.

Application to System Architecting—This section has presented several approaches commonly used to assess the value of a system architecture. While all of these approaches could theoretically be applied to system architecting, there is an implicit hierarchy in the use of these approaches that stems

Table 1. Classification of value assessment methods according to units of benefit and role of system

	Benefit is profit ($)	Benefit is performance	Benefit is utility (utils)
Direct	NPV, $E[NPV]$	End-to-end simulation	Utility theory, SVM, contingent valuation, HoQ
Indirect	ΔNPV, $\Delta E[NPV]$	OSSE	Δu

from their advantages and disadvantages. This hierarchy is illustrated in Figure 1, which provides insight into how to choose a valuation approach depending on the system architecting problem at hand.

Monetary approaches are used with top priority when applicable because they are objective and clear to communicate. NPV is the most commonly used option for systems that have a relatively clear market value, such as communications satellites. This approach has been advocated by industry, with the creation of institutions such as the Value-Driven Design Institute, and it was recently applied for example to the case of fractionated spacecraft [19].

When monetization is not possible, performance-based methods, both direct (end-to-end simulation) and indirect (OSSEs), are applied if possible, i.e., when benefit can be objectively calculated. For example, end-to-end simulation models are created for every Earth observation satellite mission at the European Space Agency, and OSSEs have been applied to a variety of ESA and NASA satellite missions [20].

If benefit cannot be objectively calculated, or if it is too expensive to apply simulation-based methods, subjective methods such as expert judgment or value-of-information can be used. For example, Ross and Hastings use MAUT in [14], and NASA uses SVMs for the assessment of their science and exploration missions [11]. Note that, in practice, since subjective metrics appear almost systematically in the system architecting process of many space science and exploration projects, expert judgment is a necessary component in an all-purpose system architecting tool. This is in opposition to a typical system design tool, which will use only objective information.

Rule-based Expert Systems

A **rule-based expert system** (RBES) is a program that uses a large number of logical rules, i.e., if-then structures, in order to solve complex problems. These rules capture the expert knowledge necessary to solve the problem at hand. RBES originated in the late sixties and early seventies at Stanford university, with a group of researchers led by Ed Feigenbaum. Feigenbaum and his colleagues were inspired by work by cognitive psychologists Newell and Simon, who around the same time published a book in which they argued that human experts store their knowledge in the brain in the form of interconnected chunks of knowledge that can be well modeled by means of logical rules [21]. Based on this discovery, Feigenbaum's group started to experiment with computer programs that tried to mimic this. The first such successful experiments were DENDRAL and MYCIN. The DENDRAL system was used to infer the composition of a material using spectral information [22]. MYCIN was a rule-based system that was capable of diagnosing a bacterial infection, and of prescribing the right kind of antibiotic for it. Using only a few hundred rules, MYCIN was able to outperform both junior and senior doctors [23].

Since DENDRAL and MYCIN, rule-based expert systems have been successfully used for a variety of applications, in many fields from different disciplines of the sciences and engineering. PROSPECTOR was utilized to predict the location of ore deposits [24]. The R1/XCON system was developed to help DEC do automatic configuration of their VAX terminals [25]. A non-exhaustive list of rule-based systems with their main functions and domains of application is provided in Table 2. Several of these examples are taken from work developed at the NASA Johnson Space Center [26].

RBES essentially combine large knowledge databases with very efficient pattern matching algorithms. Domain knowledge about how to solve a class of problems is stored in the **working memory** in the form of logical **rules**, whereas knowledge about the input problem at hand is stored in the form of **facts**. Facts use particular data structures called templates. A **template** is a list of pairs slot-value, where the slot contains the name of a certain property and the value stores the value assigned to that property. Values can be symbols (e.g. strings, scalars) or lists of symbols. For example, let us consider a template for facts of the type "launch vehicles", which contains five slots: name, performance-to-700km-SSO, payload-diameter, payload-length, and cost. One or more launch vehicle facts can be added into working memory (or asserted) using the assert command:

(assert (launch-vehicle (name LV-1) (performance-to-700km-SSO 3,000) (payload-diameter 3.0) (payload-length 5.0))).

A rule has a left hand side (LHS) that contains the if statement, and a right hand side (RHS) that contains the then statement. The LHS contains a set of conditions that take the form of facts. The RHS contains a set of actions that typically involve the modification of existing facts and the creation of new facts. These actions are executed only if the conditions in the LHS are all satisfied.

More precisely, the inference algorithm performs two actions iteratively: pattern matching all the facts with all the LHS of the rules, which yields one activation record per matching, and executing actions, which asserts and modifies facts according to the RHS of the rules in the activation records. Every time a rule is fired, the pattern matching is redone, but in a very efficient way thanks to the use of the Rete algorithm [27]. An OPM diagram of this process is illustrated in Figure 2.

Research Goals

The goal of this piece of research is to develop a methodology to assess the value of a system architecture. The methodology has several requirements:

(1) The methodology must be centered in the maximization of value delivery to stakeholders, in the most explicit and holistic way. This is in opposition to other approaches exclusively based on profit or technical performance.

(2) The methodology must be capable of handling large quantities of heuristic and subjective knowledge efficiently. This includes having data structures and algorithms that can

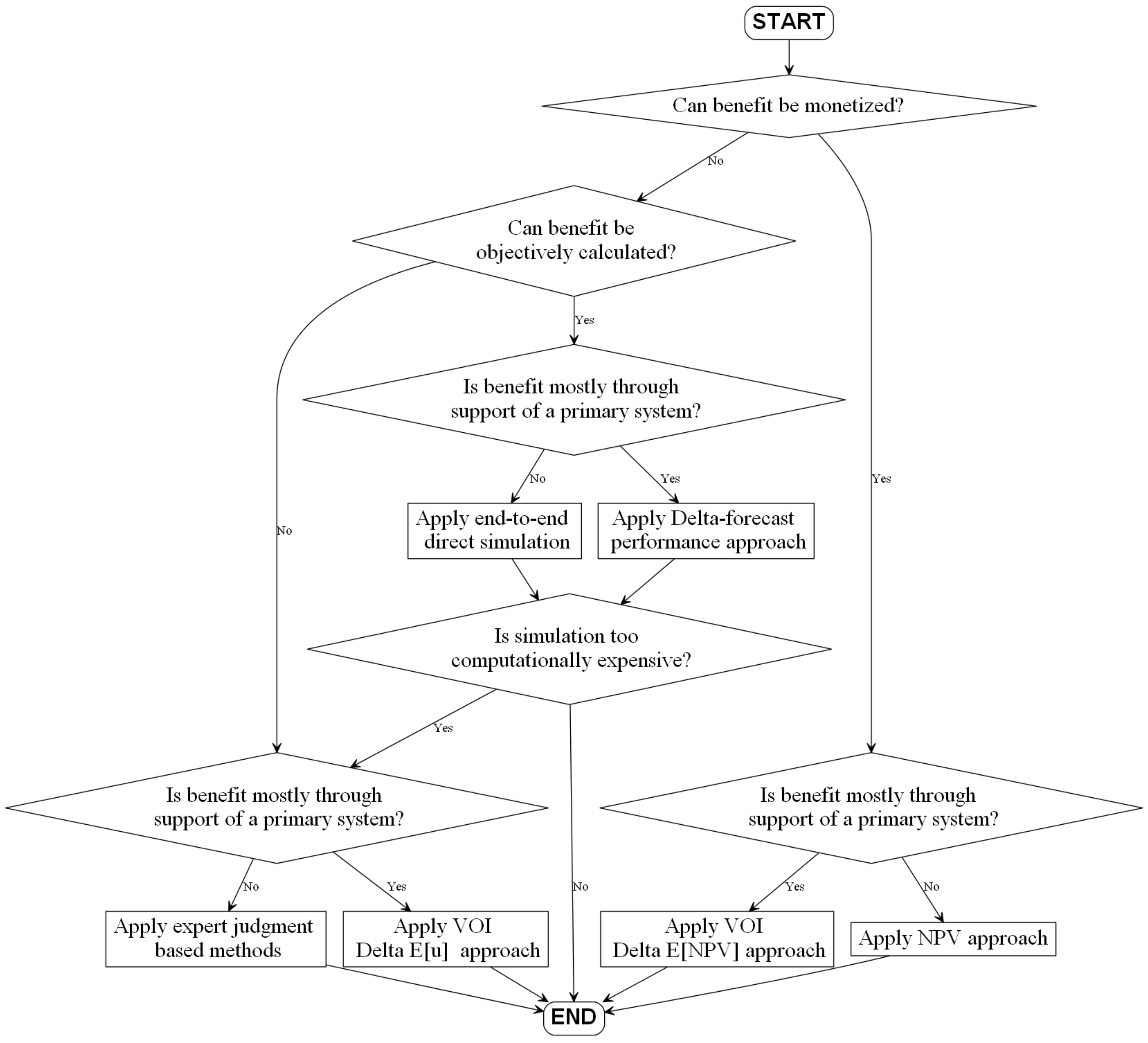
START
Can benefit be monetized?
No
Yes
Can benefit be objectively calculated?
No
Yes
Is benefit mostly through support of a primary system?
No
Yes
Apply end-to-end direct simulation
Apply Delta-forecast performance approach
Is simulation too computationally expensive?
Yes
No
Is benefit mostly through support of a primary system?
No
Yes
Apply expert judgment based methods
Apply VOI Delta E[u] approach
Is benefit mostly through support of a primary system?
Yes
No
Apply VOI Delta E[NPV] approach
Apply NPV approach
END

Figure 1. Flow diagram showing how valuation approaches are used in system architecting

Table 2. **Some examples of rule-based systems**

Name of RBES	**Main developers**	**Functions**	**Domain of application**
DENDRAL	Feigenbaum, Buchanan, et al	Inference	Mass spectroscopy, organic chemistry
MYCIN	Shortliffe, Buchanan, et al	Diagnosis, prescription	Bacterial infections, medicine
PROSPECTOR	Duda et al	Prediction	Mining, ore deposits
R1/XCON	McDermott et al	Configuration	Computer engineering
LUNAR	Woods et al	Database	Geology, chemical composition
GUIDON	Clancey et al	Teaching	Education
NAVEX	Culbert et al	Navigation	Spacecraft navigation

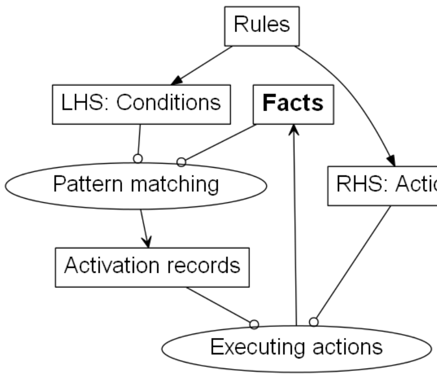

Figure 2. Structure of an RBES

effectively capture heuristic knowledge.

(3) The methodology must provide traceability of the value delivery loop. This requirement stems from the combination of large uncertainty in the architecture process and the complicated coupling and emergent behavior that is characteristic of system architecting. The goal is to support the system architect by providing not only the value of a system architecture, but also the explanations behind this value.

(4) The methodology must capture emergent behavior. Emergent behavior occurs when a combination of elements, processes, or capabilities creates a new capability that the individual elements were unable to perform. Value delivered to stakeholders usually stems from emergent behavior, which justifies the importance to model it correctly.

In this paper, a methodology is proposed that satisfies these requirements through the use of RBES. In particular, a rules engine is incorporated into the traditional objective or value function. A rule-based approach was chosen over other more complex knowledge-based systems (e.g., frame-based systems) because:

(1) Logical rules are extremely powerful to communicate expert knowledge due to their simplicity (in other words, everyone understands and if-then statement).

(2) The data structures used in logical rules, which are called templates and essentially are variations of LISP's *assoc-lists*, are ideal to capture most of the architectural decisions.

(3) Logical rules are also a natural way of modeling emergent behavior.

2. Value Assessment of System Architectures Using Rules (VASSAR)

The VASSAR methodology for value assessment of system architectures using rules is described in this section. The tool leverages from the communicating power of logical rules as data structures, from the natural recursivity of functional languages, and from the traceability and scalability of rules engines, to efficiently incorporate a large quantity of expert knowledge into the value functions.

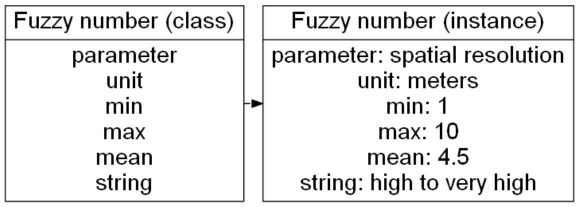

Figure 3. Fuzzy number data structure

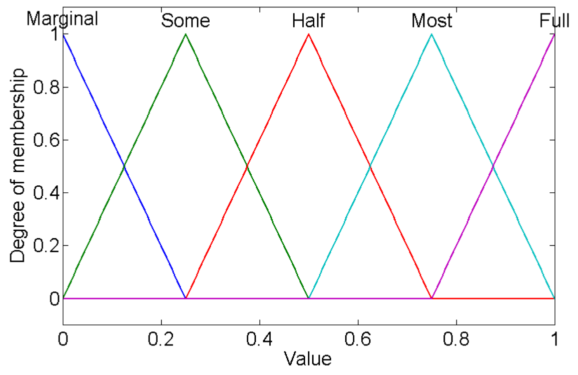

Figure 4. Membership functions for fuzzy numbers representing requirement satisfaction

Fuzzy Numbers

VASSAR uses a fundamental data structure that we call **fuzzy numbers**. Our fuzzy numbers are a special case of fuzzy sets in the sense of Zadeh [28], with triangular membership functions, and center-of-gravity "defuzzyfication". A fuzzy number consists of a parameter (string), three real numbers (min, max, mean), a unit (string), and a qualitative assessment (string). A fuzzy number fundamentally represents some magnitude with uncertainty, and it can be equivalently seen as: a) an interval, e.g.: spatial resolution = 10-100m; b) a mean value and a standard deviation spatial resolution = 55+-45 (m); c) a qualitative assessment, e.g. spatial resolution = "medium-high". The representation of a fuzzy number (class and one example instance) is provided in Figure 3.

In VASSAR, fuzzy numbers are used to represent both the cost and performance of a system architecture, and the requirements and preferences from stakeholders. In particular, the level at which a certain requirement is satisfied is encoded using a fuzzy number, according to the membership functions illustrated in Figure 4. It is important to note that 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" (e.g. spatial resolution below one meter from GEO) and "show-stopper" (e.g. no all-weather capability) scenarios.

Blackbox (input-output) model of VASSAR

At the highest level of abstraction (level 0 by convention), VASSAR simply is a collection of value functions, i.e., a mapping from the domain of architectural decisions to the domain of value, or utility to stakeholders.

More precisely, the input to VASSAR is a system architecture represented as an unordered list of pairs *decision: value*. This data structure is called *assoc-list* in the LISP programming

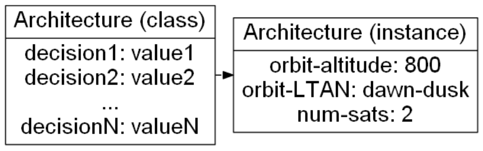

Figure 5. Architecture data structure

language. Values are symbols and therefore can contain any numeric value, string, or generic data structure including a fuzzy number. The representation of an architecture (the class and one example instance) is provided in Figure 5. The example is based on a hypothetical soil moisture remote sensing satellite mission carrying two instruments, for which we focus on three high-level decisions: the orbit altitude (400, 600, or 800km), the local time of the ascending node or LTAN (dawn-dusk, AM, or PM), and a third decision encoding whether the two instruments are flown on a single spacecraft or on two different spacecraft.

Given these decisions and options, the size of the tradespace (i.e. the number of different possible architectures) is $3 \times 3 \times 2 = 18$ architectures. Note that, in general, the size of the tradespace is given by the product of the number of options per decision, and therefore it grows exponentially with the number of decisions, where the base is the geometric average number of options per decision.

The output of the VASSAR methodology is a fuzzy number representing the value of the system architecture defined as the ability of the system architecture to satisfy stakeholder requirements. In the case of our example mission, an architecture could for example be assigned a fuzzy number "80-90%: fully satisfies all critical requirements and most non-critical requirements". This value would be accompanied by detailed explanations (plain text) of how critical requirements are fully satisfied, which non-critical requirements are only partially satisfied or completely missed, and why, e.g.: "the architecture fully satisfies hydroclimatology requirements for soil moisture data products thanks to the combination of active and passive datasets using disaggregation scheme", or "the architecture misses weather hydrometeorology requirements for soil moisture data products because of insufficient spatial resolution".

The blackbox diagram of VASSAR using these two data structures is illustrated in Figure 6.

Level-1 model of VASSAR

The theory of system architecture teaches us to architect a system by going from stakeholder requirements to a functional decomposition of the system and then to a decomposition in terms of form through the definition of an integrated concept. Assessing the value of an architecture can be seen as performing the opposite process of architecting. Thus, as a first approximation, the VASSAR methodology assesses the value of a system architecture by computing the capabilities of a system architecture and then comparing the capabilities with the stakeholder requirements.

Our use of the word capabilities deserves further explanations. Systems provide value to stakeholders because they are capable of conducting certain functions at the required level of performance, i.e., capability is function at performance. Some authors use the term capabilities to designate the function, without the performance side (e.g., "this mission has the capability of measuring soil moisture"). In this paper, the term capabilities includes both the function and the performance (e.g., "this mission has the capability of measuring soil moisture with a spatial resolution of 10km, an accuracy of 5%, and a temporal resolution of 3 days").

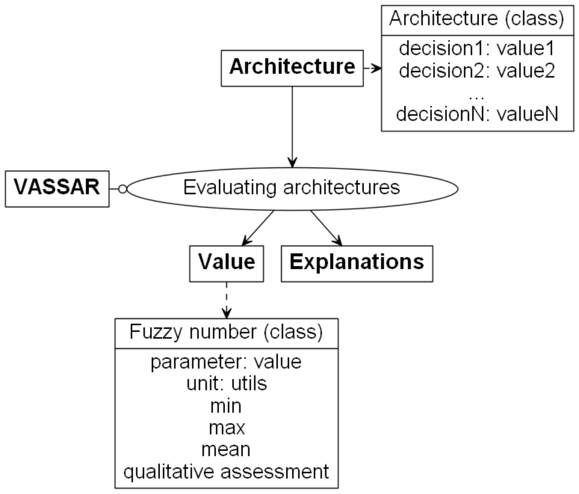

Figure 6. Blackbox model of VASSAR

It is important to note that the process of computing capabilities is extremely non-linear and non-sequential, as it captures the emergent behavior that leads to creation of value. New capabilities are created from combinations of existing capabilities, and higher level performance attributes can only be computed as the required level performance attributes become available. Individual rules can be written to compute each performance attribute, which has at least two advantages: a) increased flexibility, because the way an attribute is computed can be changed without changing anything else in the program; b) increased scalability, because a new performance attribute can be added without affecting the rest of the code.

At the first level of decomposition, the VASSAR methodology consist of three major processes that occur in sequence: first, the capabilities of an architecture (function and performance) are computed from the architectural decisions using domain knowledge; second, requirement satisfaction is computed by comparing capabilities with stakeholder requirements; third, individual requirement satisfaction is aggregated into one or a handful of metrics that represents the value of the system architecture, using information about stakeholder preferences.

In parallel of these three processes, an explanation facility keeps track of the entire value delivery loop, from architecture to value through capabilities and requirement satisfaction, and constructs a set of explanations that are provided to the user with the value fuzzy number. This process is illustrated in Figure 7.

Level-2 model of VASSAR

In this section, the three steps of the VASSAR methodology are described in more detail.

Step 1: Compute Capabilities—Capabilities can usually be computed from architectural decisions using domain knowledge in the form for example of physics laws. However, the

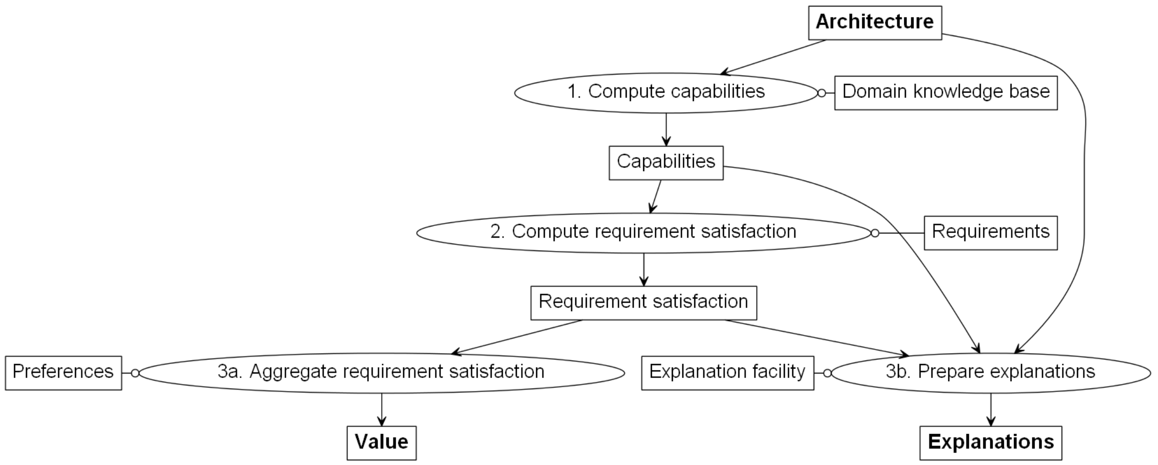

Figure 7. Level 1 model of VASSAR

process of computing capabilities from decisions and domain knowledge typically is highly non-linear and non-sequential. In other words, capabilities are computed as the information required to compute them becomes available, and they must be recomputed every time this information changes. It is precisely for this reason that rule-based engines are naturally fit for the process of computing capabilities from decisions. The system architect can focus on the domain-specific code, and let the RBES take care of the execution flow control, including all the required iteration.

The following example better illustrates how capabilities are computed from decisions. In our moisture mission, orbit altitude (a decision) affects both temporal and spatial resolution of the data products (capabilities). LTAN (another decision) sets the time of the day at which observations are made at different latitudes, which impacts illumination conditions, and sensitivity to soil moisture due to diurnal cycles (capabilities). LTAN also impacts cost through the power and thermal budgets of the spacecraft. The decision to fly the two instruments on a single spacecraft or on separate spacecraft will affect both science and cost. If flown on a single spacecraft, the instruments can share a dish, and data cross-registration will be easier, but they are also forced to share resources such as power and data rate, potentially resulting in a decreased duty cycle for one or both of the instruments. If flown on dedicated spacecraft, two spacecraft need to be designed, implemented, tested, launched, and operated, but the design of each bus is simpler and can be tailored to the specific instrument. Individual instrument data rates are higher, but synergies between instruments are harder to exploit. This complicated network of interactions is better modeled using RBES than using traditional procedure-based models, for which iteration would be required. We show later how a few rules suffice to capture most of the variation of value across this tradespace.

The process of computing capabilities can be decomposed into five main processes, as illustrated in Figure 8: a) asserting architectural facts; b) inheriting attributes from upper to lower levels of the hierarchy of form decomposition; c) computing basic capabilities; d) computing emergent capabilities; e) computing performance.

From these five processes, the assertion of architectural facts always occurs first. The process takes an architecture and asserts all the corresponding system, subsystem, and component facts. Note that VASSAR uses a 3-level form decomposition nomenclature (system, subsystems, and components) following INCOSE's recommendation [29]. For example, in the case of our mission, the mission itself is the system, the spacecraft are the subsystems, and the instruments are the components. More than three levels are rarely required in the context of system architecture. Fewer (two) levels may be enough for simpler systems.

Subsystem and component facts created through manifest rules typically have many slots empty, because not all properties are directly set by architectural variables. These slots are filled out in the second step, called attribute inheritance. There are three main types of attribute inheritance: a) direct inheritance from upper levels of the architectural hierarchy, e.g., the orbit of an instrument is inherited from the orbit of its parent spacecraft; b) direct inheritance from a database of fixed parameters such as components characteristics, e.g., the angular resolution of an instrument; c) indirect inheritance, in which attributes are calculated using several other attributes, e.g., the cross-track ground spatial resolution of a side-looking instrument can be computed from the orbit altitude h, its off-nadir angle θ_{off}, and its angular resolution $\Delta\theta$:

$$\Delta x_{ct} = h(\tan(\theta_{off} + \frac{\Delta\theta}{2}) - \tan(\theta_{off} - \frac{\Delta\theta}{2})) \qquad (1)$$

The third step is the assertion of an initial set of capabilities 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, in the case of our example mission, one of these rules can read, in natural language: IF there is an L-band radiometer, THEN assert a capability to measure soil moisture. At this point the performance attributes of this measurement (i.e., its accuracy, temporal resolution, spatial resolution, and so forth) are not known.

The two remaining processes of computing capabilities, namely computing emergent capabilities and performance, occur simultaneously: 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. Convergence criteria can be included in the rules to break potentially infinite loops in the execution flow as needed (e.g., in the design of a spacecraft, the sizing of the power subsystem depends on the power requirements of the ADCS, which depend on the mass of the spacecraft and therefore on the mass of the power subsystem, thus creating a loop that is resolved by introducing a convergence criteria on the in dry mass between consecutive iterations).

Performance rules are similar in structure to attribute inheritance rules, but they concern capability facts instead of architectural facts. Performance attributes of capability facts are sometimes inherited or directly copied from the corresponding component facts. For example, the ground spatial resolution of a measurement taken by a certain instrument is given by the ground spatial resolution of the instrument. Other performance attributes must be calculated from combinations of available attributes.

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. For example, in the case of our example mission, disaggregation schemes allow to combine the high resolution, low accuracy dataset from the radar with the low resolution, high accuracy dataset from the radiometer to obtain a medium-high resolution, high accuracy dataset [30]. A rule capturing this emergent behavior could read, in natural language: IF there is a soil moisture dataset with high spatial resolution and low accuracy, and another soil moisture dataset with low spatial resolution and high accuracy, and the two datasets are cross-registered, THEN assert a new soil moisture dataset with medium spatial resolution and high accuracy. Note the importance of the condition that the two datasets be cross-registered in order for the new capability to be asserted: the disaggregation scheme may be hard to apply to architectures in which the two instruments are not on the same spacecraft.

Attribute inheritance, performance, and capability rules use two types of mappings between variables: a) equations; b) heuristics in the form of case-based mappings, such as “if LTAN is DD, then sensitivity to soil moisture is high”. Equations are used whenever it is possible, as they allow exact reasoning. Heuristics are used: a) in order to abstract out particularly expensive systems of equations when results are approximately known for regions of the input space; in this context they act as the equivalent of response surfaces in purely discrete domains; b) to introduce subjective capabilities, as logical rules are often the most natural way of expressing subjective knowledge, e.g.: “if the architecture is composed of small satellites, then it is more likely to be robust to political instability”.

In the VASSAR methodology, the system architect can manually enter one-of-a-kind capability rules. However, a spreadsheet-based user interface was implemented for rapid coding of attribute inheritance rules, and repetitive capability rules.

Step 2: Compute Requirement Satisfaction—Capabilities and satisfaction are not synonyms. Indeed, having more capabilities does not always linearly translate into more value (i.e. more stakeholder satisfaction). For example, the real benefit of decreasing the revisit time in a constellation of satellites

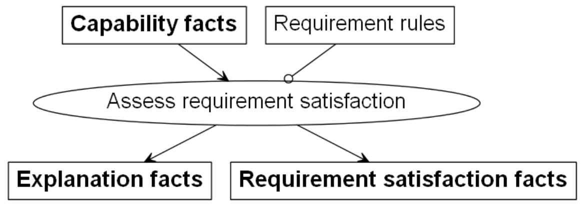
Capability facts
Requirement rules
Assess requirement satisfaction
Explanation facts
Requirement satisfaction facts

Figure 9. Level 2 VASSAR, step 2: Computing satisfaction

does not increase linearly as revisit time decreases. Instead, the relationship between capabilities and requirement satisfaction may exhibit discrete steps (e.g., the ability to see shorter time-scale phenomena such as cloud formation), saturation effects (e.g., we get no marginal benefit from decreasing temporal resolution below a certain level because we are limited by the performance of current algorithms), and non-linear continuous multiplicative envelopes (e.g., a concave risk aversion envelope capturing decreasing marginal benefit). In the VASSAR methodology, the relationship between capabilities and satisfaction is expressed in the form of requirement satisfaction rules that capture these discrete steps, saturation effects, and non-linear continuous envelopes (see Figure 9).

Requirement satisfaction rules assess how well each particular requirement is satisfied by the system architecture. Mathematically, they map the multidimensional space of capabilities/performance to the one-dimensional space of satisfaction for one particular requirement. We call these mappings satisfaction functions. Note that there are as many satisfaction functions as there are requirements.

The simplest possible satisfaction function is an identity satisfaction function. This implies a perfectly linear relationship between capabilities/performance and satisfaction. For example, the value of an imaging satellite can be summarized in a performance attribute of #images per day, so that a system A that produces twice as many images per day as a system B, is also providing twice as much value to its stakeholders. However, this is rarely the case in practice.

In general, a requirement takes the form of a capability plus a number of conditions on performance attributes, e.g.: our mission shall measure soil moisture with a spatial resolution of 10km or better and a temporal resolution of 2-3 days. In this case, the requirement provides a single threshold value for each performance attribute (i.e., spatial resolution and temporal resolution). This type of requirements leads to a step satisfaction function: either the requirement is fully satisfied, if all the performance attributes are at or above the threshold values; or it is not satisfied at all, in any other case.

Step functions are very simple to implement as they only require one rule per requirement. Moreover, many requirements in practice consist of a single threshold value. However, one rule is arguably not sufficient to capture complex satisfaction functions. On the other side of the spectrum, we could imagine using a continuous function to map the multi-dimensional performance attribute space to the one-dimensional requirement satisfaction space. Inferring the shape of such continuous satisfaction functions is challenging in general, and would require the use of formal elicitation methods such as utility theory or AHP [13].

A possible compromise between step functions and arbitrary continuous functions is using discrete curves with a handful

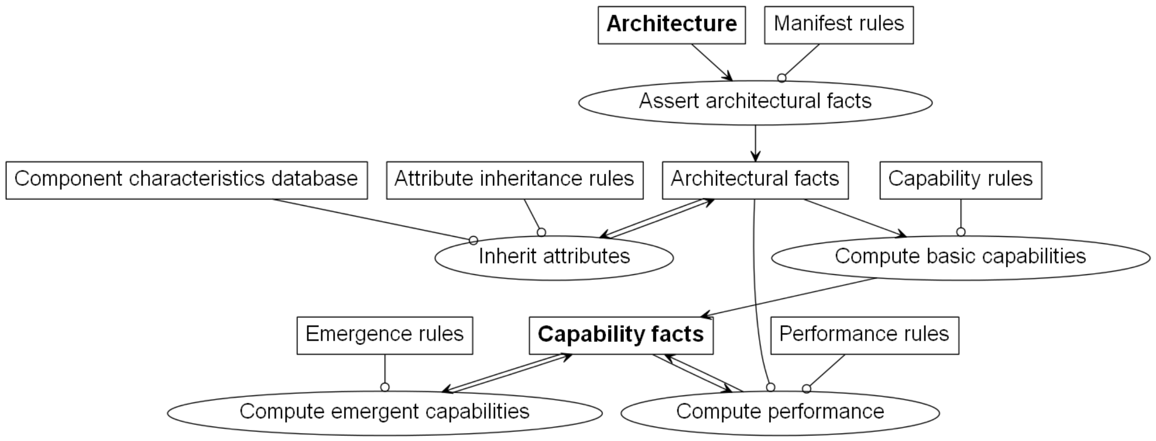

Figure 8. Level 2 VASSAR, step 1: Computing capabilities

of steps. In fact, many system requirement documents express requirements using a target value and a threshold value. This suggests the use of bi-level satisfaction functions. Bi-level or multi-level satisfaction functions are fully defined by a set of thresholds and a set of corresponding satisfaction values.

Note that, mathematically, step functions have one-dimensional domains. However, as stated before, the domain of satisfaction functions (i.e., the capabilities domain) is in general multi-dimensional, as a particular requirement may look at different attributes simultaneously (e.g., spatial resolution and temporal resolution). Hence, intervals of constant values in one-dimensional satisfaction functions become regions or hyperplanes of constant values in multi-dimensional satisfaction functions.

Multi-dimensional satisfaction functions can in principle be avoided by using different rules (and satisfaction functions) for expressing the requirements concerning each individual attribute. For example, in the preceding example, three different rules and satisfaction functions could be used for accuracy, spatial resolution, and temporal resolution. However, the combination of the satisfaction values referring to individual performance attributes is still necessary and simply deferred to the value aggregation step. Furthermore, this combination process may itself be very challenging. Simple solutions, such as computing the product of individual attribute satisfaction to obtain subobjective satisfaction, may be inappropriate due to the coupling between individual requirements. For example, the threshold for spatial resolution may depend on the temporal resolution provided by the system, and vice-versa.

These problems are best avoided by using multi-dimensional satisfaction functions that take into account several attributes at the same time. The drawback of this approach is that it could potentially require many steps (and thus many rules) to cover the same cases as the previous approach. For example, assuming a bi-level satisfaction function for temporal resolution, spatial resolution, and accuracy, leads to 8 possible performance scenarios. A multi-dimensional satisfaction function would thus require in principle 8 rules, one per scenario. The one-dimensional satisfaction function would require 3 rules (one per attribute), plus as many rules as needed in order to account for all the trade-offs between attributes. In practice, the multi-dimensional approach may be easier to implement for reasonable numbers of attributes, where as the single-dimension approach is almost necessary for many attributes. Note that in the multi-dimensional it is sometimes possible to use expert knowledge in order to cover only the subset of these 8 scenarios that are actually plausible.

An important remark concerning this methodology is that requirement satisfaction uses case-based reasoning: one particular rule only covers one particular case (e.g., the nominal case in which spatial resolution, temporal resolution, and accuracy 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 system architecture provides a null (0%) satisfaction of that requirement. In a hypothetical case where there was only this requirement for the system architecture, the tool would come up with a value of 0.0 for this system architecture. 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 full satisfaction rule plus a handful of partial satisfaction rules can cover all realistic cases for a given set of architectural decisions.

Note that multi-level step functions can approximate any continuous or discrete satisfaction function at the desired level of accuracy by simply increasing the number of steps. In particular, any of Messac's canonical satisfaction function can be modeled [31].

In the VASSAR methodology, the system architect can enter one-of-a-kind requirement satisfaction rules. However, a spreadsheet-based user interface was implemented for rapid coding of multi-step satisfaction functions, which have all the same structure.

Step 3a: Aggregate Requirement Satisfaction—The preference domain is hyperdimensional because there is at least one dimension per requirement, and in a complex system there can easily be several hundreds or even thousands of requirements. Dominated architectures could easily be eliminated by means of Pareto analysis, but this is unlikely to reduce the size of the tradespace by much due to the large number of requirements. Hence, the problem remains of choosing between architectures that have very different requirement

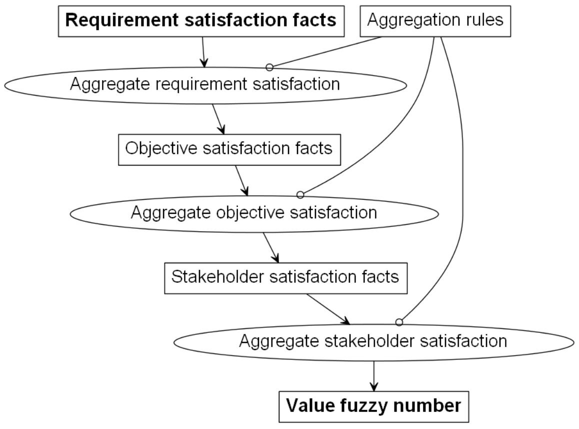
Requirement satisfaction facts
Aggregation rules
Aggregate requirement satisfaction
Objective satisfaction facts
Aggregate objective satisfaction
Stakeholder satisfaction facts
Aggregate stakeholder satisfaction
Value fuzzy number

Figure 10. Level 2 VASSAR, step 3a: Aggregate satisfaction

satisfaction sets, but seem globally equally attractive. In practice, 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 stakeholder preferences.

Hence, the last step of the methodology is the reduction of the dimensionality of the preference domain through aggregation of requirement (i.e., subobjective) satisfaction into objective satisfaction, and then into stakeholder satisfaction. This two-step aggregation process requires the elicitation of a list of objectives and subobjectives from different stakeholders, and reduces the dimensionality of the preference domain (i.e., the number of metrics) from N_{req} to N_{sh}.

The number of metrics can be further reduced from N_{sh} to 1 if the relative importance of stakeholders to a central stakeholder is introduced. The relative importance of stakeholders can be formally computed by applying quantitative stakeholder analysis techniques (e.g. stakeholder networks [32]). 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 Figure 10.

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 stakeholder 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 [33], [34]). 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 these 4 objectives are satisfied."

In the VASSAR methodology, the system architect can manually define one-of-a-kind aggregation functions. Weighted averages can be rapidly coded in from a spreadsheet-based user interface.

Step 3b: 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 delivery loop. Explanation rules keep track of the entire value delivery loop by extracting information from architectural facts, capability facts, requirement satisfaction facts, objective satisfaction facts, and stakeholder satisfaction facts, as illustrated in Figure 11.

Table 3. **Decisions and range of options for the case study**

Decision	Range of values
#planes	1-4
#sats per plane	1-4
Orbit altitude	[400; 800]km in steps of 100km
Orbit inclination	polar or SSO
Orbit RAAN	Dawn-dusk or AM
Payload selection	Any combination of the 5 instruments described in the payload description subsection (31 combinations)
Spacecraft architecture	Any partition of the payload into spacecraft (between 1 and 52 combinations, depending on number of instruments)

3. CASE STUDY

We selected a hypothetical soil moisture remote sensing satellite mission as the case study to show the applicability the VASSAR methodology. This hypothetical mission would combine instruments that are similar to real instruments flown or to be flown in NASA and ESA missions. References to these missions will be made as required. This example was designed to illustrate different features of the methodology:

(1) A large portion of the value it delivers is scientific value, which contains elements of subjectivity, which are hard to model in traditional design tools;

(2) it shows clear examples of emergent behavior that actually drives an important part of the value delivery;

(3) it has the right level of complexity, as it is complex enough to yield non-trivial results, but simple enough to avoid the need to provide excessive background.

Payload description

For this example we consider five candidate instruments: a) an L-band polarimetric radiometer (LRADIO), such as the one on NASA's SMAP mission; b) an L-band synthetic aperture radar (LSAR), such as the one on the SMAP mission; c) an X-band polarimetric radiometer (XRADIO), such as NPOESS/CMIS; d) an IR multispectral radiometer (IR) such as NPOESS/VIIRS; e) a hypothetical P-band polarimetric SAR (PSAR), such as the one proposed for ESA's BIOMASS. The characteristics (mass, power, data rate, performance) that we assumed for these instruments are provided in Appendix A.

Architecture tradespace

For this example, we chose to represent an architecture as the set of decisions laid out in Table 3. The set of possible architectures is given by the Cartesian product of the ranges of values in Table 3, which consists of 3,636 architectures.

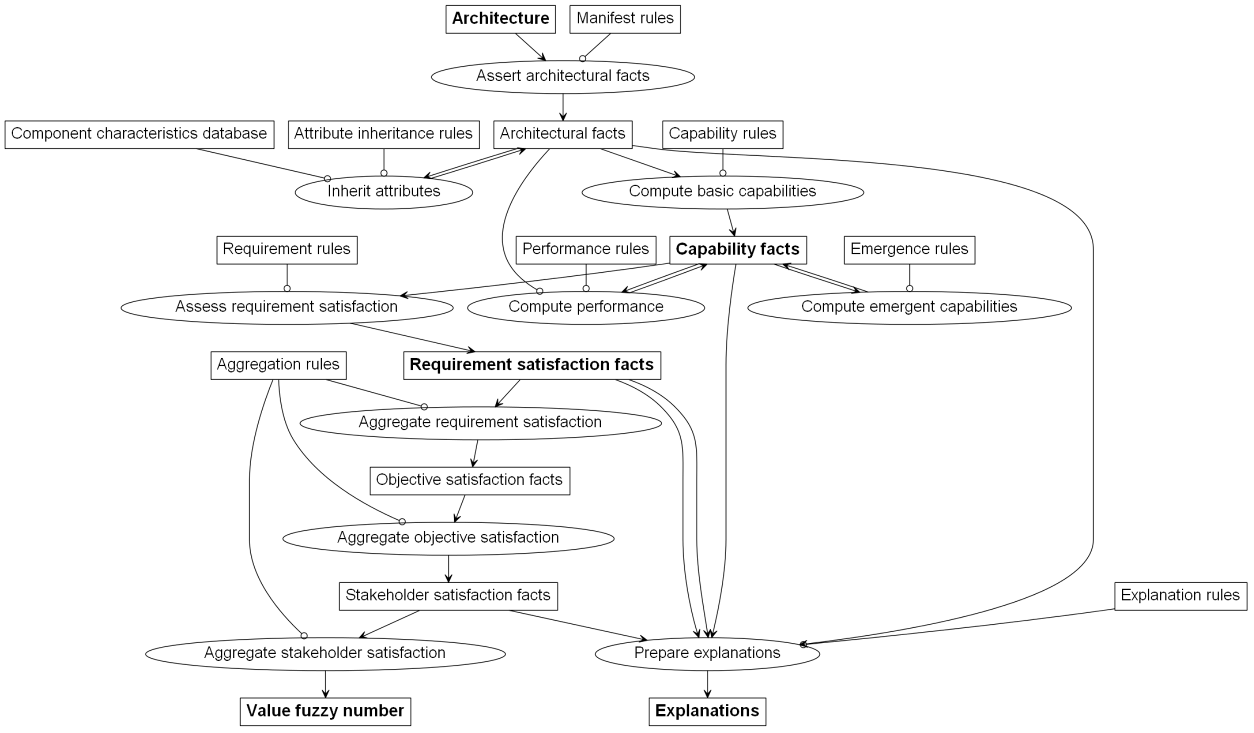
Architecture
Manifest rules
Assert architectural facts
Component characteristics database
Attribute inheritance rules
Architectural facts
Capability rules
Inherit attributes
Compute basic capabilities
Requirement rules
Performance rules
Capability facts
Emergence rules
Assess requirement satisfaction
Compute performance
Compute emergent capabilities
Aggregation rules
Requirement satisfaction facts
Aggregate requirement satisfaction
Objective satisfaction facts
Aggregate objective satisfaction
Stakeholder satisfaction facts
Explanation rules
Aggregate stakeholder satisfaction
Prepare explanations
Value fuzzy number
Explanations

Figure 11. Level 2 VASSAR, step 3b: Prepare explanations

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 et al [32]. The specific objectives of each panel, as well as their relative ranking, are provided in Appendix B.

Capability rules

Nominal instrument capabilities are presented in Appendix A. 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 NIR instruments (namely the corresponding channels of the IR 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.

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 land-cover status.

Multispectral measurements—IR and MW 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.

Results

The set of 3,636 architectures was evaluated using VASSAR. The benefit portion is based on the set of requirements from Appendix B. The cost portion is based on the cost model described in Appendix C. The point science scores and lifecycle cost estimates for all these architectures are shown on Figure 12. These numbers are obtained from defuzzyfying the fuzzy

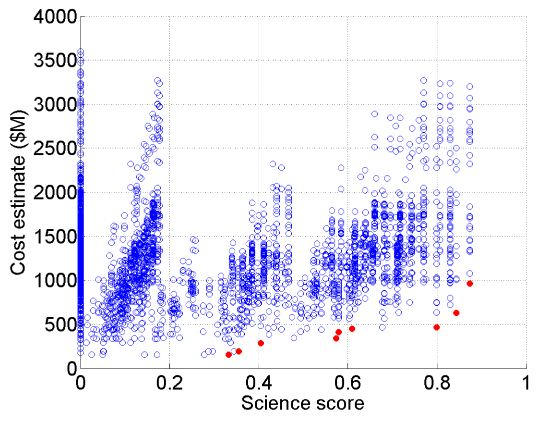

Figure 12. Expected science vs expected lifecycle cost for 3,636 architectures (uniform weights)

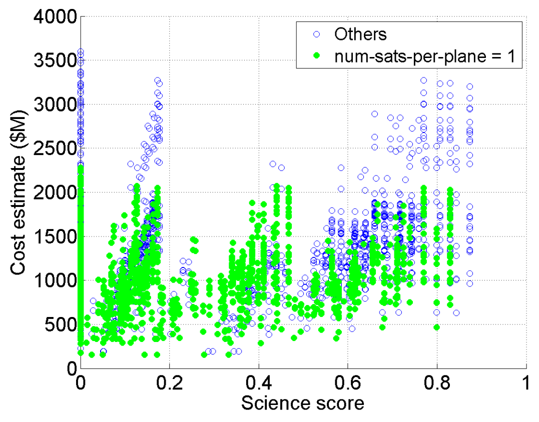

Figure 13. Architectures with one satellite per plane highlighted on the tradespace

values. Non-dominated architectures are highlighted in red.

Tradespace analysis—The main goal of any system architecting tool is to gain insight into the "shape" of the tradespace, what the main trades are, whether there are families of architectures, and so on. We observe several features just by looking at the tradespace in Figure 12:

(1) 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.

(2) 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.

(3) 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.

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:

(1) 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 4.

(2) 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 Figure 12 by clicking on them, as shown in Table 5. 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.

The tool can also highlight the regions of the tradespace that correspond to a particular combination of architectural decisions. For example, Figure 13 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.

(3) 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 between $[0.01; 0.2]$, the tool automatically detects that most of the architectures in this region have polar, non-SSO orbits, as shown in Figure 14.

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 RBES. It is important to note that conceptually, we are using fuzzy numbers for two slightly 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 is provides less information than treating it with probability distribution functions (pdf), because we only get information about the boundaries of the pdf. In this example, we are using fuzzy numbers to

Table 4. 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)

Table 5. Details of non-dominated architectures from Figure 12 (uniform weights)

Arch#	Payload	Instr allocation	h (km)	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 LSAR	[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

represent statistical uncertainty in cost, and non-statistical uncertainty in science. Figure 15 shows the magnitude of the uncertainty for the architectures on the Pareto frontier of Figure 12. The sources of uncertainty for cost are mostly the standard errors from the CERs 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.

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 CERs are similar in magnitude [35]. However, uncertainty in science is not homogeneous because a requirement fully satisfied 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.

Sensitivity Analysis—There are several ways of conducting a sensitivity analysis in VASSAR:

1) by simply rerunning the tool with different sets of parameters. In this section we provide two examples of sensitivity analysis: one concerning the capabilities of the P-band SAR 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 P-band SAR to provide useful measurements of soil moisture, which has not yet been proven. The appeal of P-band SAR 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 [36]. Hence the question of whether this instrument will be able of producing useful measurements of

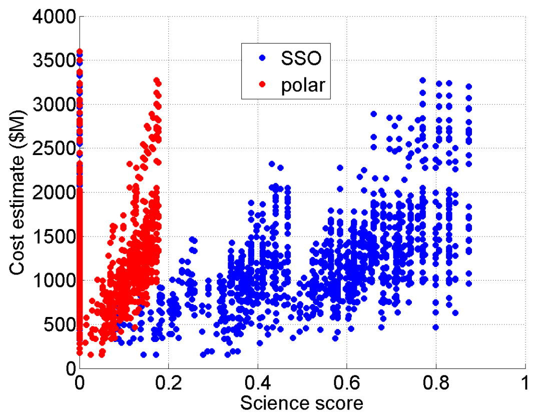

Figure 14. Effect of orbit inclination on science and cost

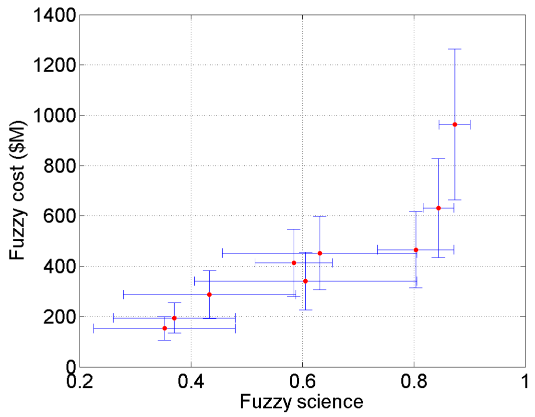

Figure 15. Fuzzy Science vs fuzzy lifecycle cost for non-dominated architectures (uniform weights)

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 6. We note several changes with respect to Table 5. 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 7. 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

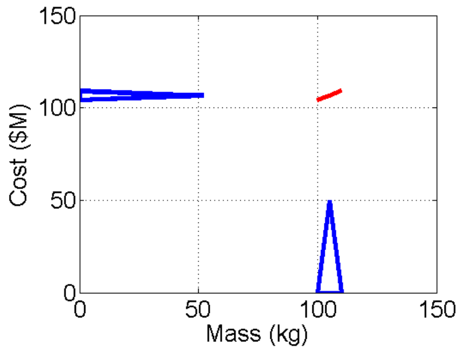

Figure 16. Propagation of uncertainty in payload mass to payload cost.

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.

More advanced variants of this option allow to compute the threshold value for a parameter that makes the non-dominated set change. For example, the tool determined in this case that for $\frac{w_{ECO}}{w_i} < 2.59$, the best architectures remain similar to those presented in Table 5, whereas for $\frac{w_{ECO}}{w_i} \geq 2.59$, the best architectures switch to those presented in Table 7. Some decision makers are more comfortable with this kind of result than plain results, because they feel that they can evaluate the certainty with which the statement $\frac{w_{ECO}}{w_i} < 2.59$ is true more easily than they can assign weights to the different stakeholder panels.

2) 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 Figure 16. 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.

3) by conducting a local search around a particular region or architecture and looking at how figures of merit 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 instruments is added to or removed from the payload, or the LTAN is changed from DD to AM. We used VASSAR to look around the architecture #669, highlighted in Table 5. Eleven architectures were thus automatically enumerated and evaluated. They are shown in Figure 17. 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.

Table 6. **Details of non-dominated architectures when P-band SAR cannot measure soil moisture (uniform weights)**

Arch#	Payload	Instr allocation	h (km)	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

Table 7. **Details of non-dominated architectures when ecosystems panel is 3 times as important as the others**

Arch#	Payload	Instr allocation	h (km)	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
810	LRADIO	$[1]$	800	SSO DD	1

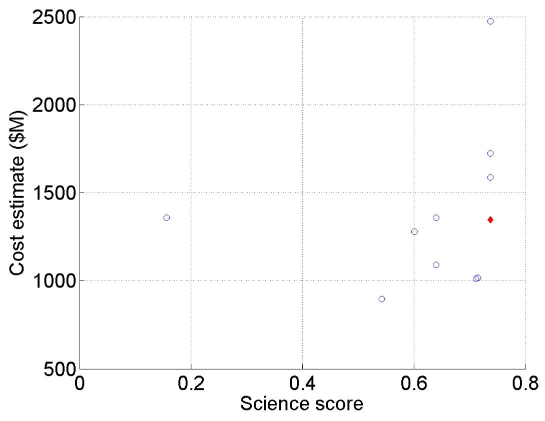

Figure 17. Local search around architecture #669, highlighted on Table 5

4. Conclusion

This paper has presented a methodology to assess the value of a system architecture using RBES. Such methodology can be used in the context of automatic architectural tradespace exploration to develop the part of computational tools that automatically evaluates the architectures being generated. An example of such tool was shown in the context of Earth Observing Satellite Systems (EOSS). 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.

Advantages and disadvantages of VASSAR

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

(1) (+) 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.

(2) (+) 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, simply because it is the way in which the human mind works.

(3) (+) The use of RBES 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.

(4) (+) VASSAR has some degree of commonality with the current trend to have 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.

(5) (+) 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.

(6) (+) VASSAR uses a functional programming language, which facilitates the design of recursive algorithms, which are at the core of modeling emergence, which in turn is the origin of value.

(7) (-) Logical rules are not suited to express all types of expert knowledge.

(8) (-) Traceability has a computational price, which is only worth it if the knowledge base is likely to evolve.

(9) (-) RBES may be slow when the number of rules is very large.

(10) (-) There is an up-front cost to develop the RBES and import the expert knowledge (e.g. lessons learned) into it.

Next steps

VASSAR is a methodology to develop RBES with a specific purpose: evaluating system architectures in the context of automatic architectural tradespace exploration. Using this methodology, a tool was created to evaluate architectures in the context of Earth Observing Satellite Systems. Some of the next steps concern the methodology itself, while others concern this tool.

Concerning the methodology, the extension of VASSAR's explanation facility with more advanced features requires the incorporation of a machine learning layer on top of the RBES. This type of hybrid AI tool is seen as a potentially fruitful area of research. Furthermore, this paper focused on the use of RBES for the valuation portion of the system architecting process. RBES 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 RBES [37]. This framework is continuously being improved with more efficient algorithms of enumerating certain classes of architectures. Finally, the current VASSAR implementation is restrictive in terms of representation of architectures, as they need to be represented as lists of pairs decision-value. A more advanced version of the tool could allow importing architectures expressed in more powerful representation tools such as SysML or OPM.

Concerning the tool, the knowledge base can be improved in several points, but not all improvements would have the same impact on the fidelity of the tool. For instance, it is unlikely that a more accurate ΔV budget would result in very different results. Two examples where the improvement would yield a high return in modeling fidelity are discussed below.

First, the spacecraft design algorithm could benefit from a higher-fidelity configuration module that accounts for the position of the main spacecraft components and provides more accurate estimations of the dimensions and moments of inertia of the spacecraft.

Second, the cost model uses cost estimating relationships (CERs) that are old and have very large standard errors. Higher fidelity cost models could be used instead of these CERs.

Table 10. 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%
Health	HEA	Human health	20%

Table 11. Weather panel objectives

Objective	Description	Value
WEA1	Initialization of NWP models	60%
WEA2	River forecast streamflow models	20%
WEA3	River forecast flash flood models	20%

APPENDICES

A. INSTRUMENT CHARACTERISTICS

The characteristics of the 5 instruments used in the example are provided in Tables 8 and 9. These characteristics are based on real instruments from the SMAP, NPP and BIOMASS missions. However, these real instruments are used as guidelines and our hypothetical instruments are not meant to represent the real instruments.

B. STAKEHOLDER REQUIREMENTS

The information used in the case study for the stakeholders and their hierarchy of requirements are shown below in Tables 10 to 15 and Figure 18.

C. COST MODEL

The cost model used in the case study is a rule-based cost model largely based on Larson and Wertzs Space Mission Analysis and Design (SMAD) [35]. The first level decomposition of lifecycle cost is given in Figure 19.

Payload cost is based on the NASA Instrument Cost model

Table 12. Climate panel objectives

Objective	Description	Value
CLI1	Boundary conditions for climate models	80%
CLI2	Ocean thermohaline circulation	20%

Table 13. Ecosystems panel objectives

Objective	Description	Value
ECO1	Net carbon flux in boreal landscapes	75%
ECO2	Ocean thermohaline circulation	25%

Table 8. **Instrument characteristics**

	LRADIO	LSAR	XRADIO	IR	PSAR
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**

	LRADIO	LSAR	XRADIO	IR	PSAR
Soil moisture	Highest accuracy, low spatial resolution	Low accuracy, high spatial resolution	Medium accuracy, low spatial resolution	Lowest accuracy, high spatial resolution	Medium accuracy, high spatial resolution
Freeze-thaw state		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	High accuracy, low spatial resolution		Low accuracy, low spatial resolution
Precipitation rate			High accuracy, low spatial resolution		
Ocean salinity	High accuracy, low spatial resolution	Medium accuracy, high spatial resolution			Low accuracy, low spatial resolution

Table 14. **Water panel objectives**

Objective	Description	Value
WAT1	Estimation of runoff-EVT	67%
WAT2	Estimation of precipitation	11%
WAT3	Snow and cold land processes	11%
WAT4	Sea Ice cover	11%

Table 15. **Applications panel objectives**

Objective	Description	Value
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%

[38]. Bus cost is based on the parametric relationships provided in SMAD ([35]). Since these parametric 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 Figure 20.

Once the mass, power, and delta-V budgets have been calculated, the different components of lifecycle cost shown in Figure 19 can be computed using the cost estimating relationships in SMAD.

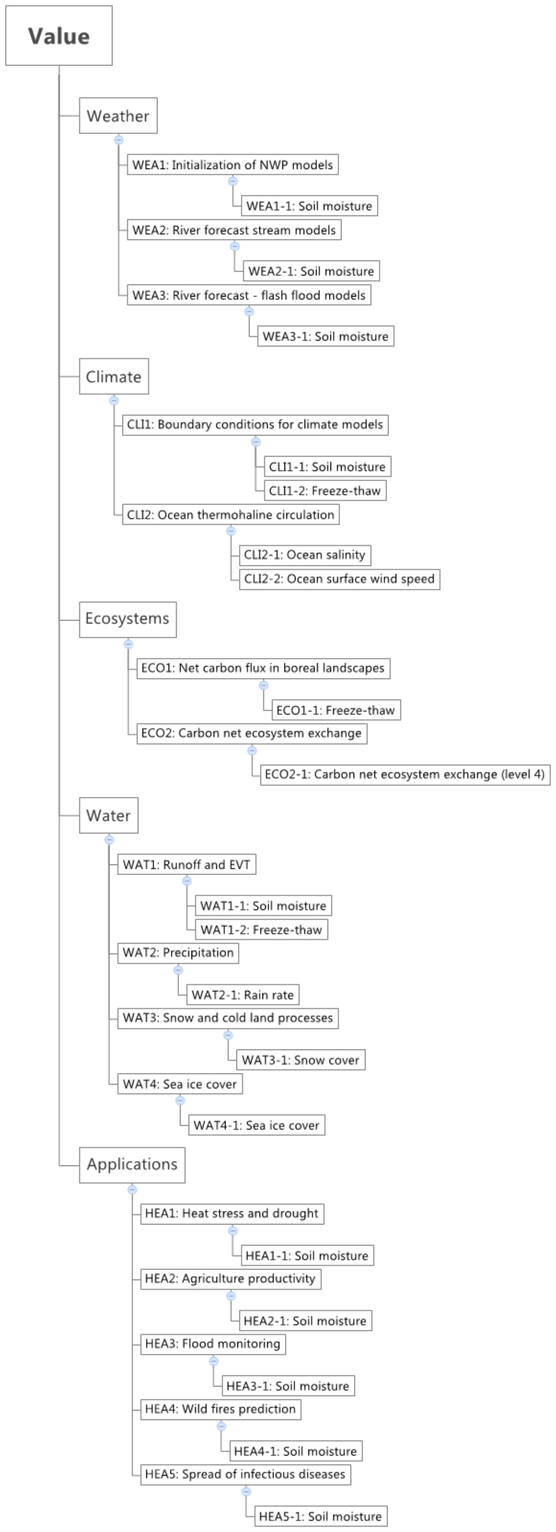
Value
Weather
WEA1: Initialization of NWP models
WEA1-1: Soil moisture
WEA2: River forecast stream models
WEA2-1: Soil moisture
WEA3: River forecast - flash flood models
WEA3-1: Soil moisture
Climate
CLI1: Boundary conditions for climate models
CLI1-1: Soil moisture
CLI1-2: Freeze-thaw
CLI2: Ocean thermohaline circulation
CLI2-1: Ocean salinity
CLI2-2: Ocean surface wind speed
Ecosystems
ECO1: Net carbon flux in boreal landscapes
ECO1-1: Freeze-thaw
ECO2: Carbon net ecosystem exchange
ECO2-1: Carbon net ecosystem exchange (level 4)
Water
WAT1: Runoff and EVT
WAT1-1: Soil moisture
WAT1-2: Freeze-thaw
WAT2: Precipitation
WAT2-1: Rain rate
WAT3: Snow and cold land processes
WAT3-1: Snow cover
WAT4: Sea ice cover
WAT4-1: Sea ice cover
Applications
HEA1: Heat stress and drought
HEA1-1: Soil moisture
HEA2: Agriculture productivity
HEA2-1: Soil moisture
HEA3: Flood monitoring
HEA3-1: Soil moisture
HEA4: Wild fires prediction
HEA4-1: Soil moisture
HEA5: Spread of infectious diseases
HEA5-1: Soil moisture

Figure 18. Hierarchy of stakeholder needs for the soil moisture case study

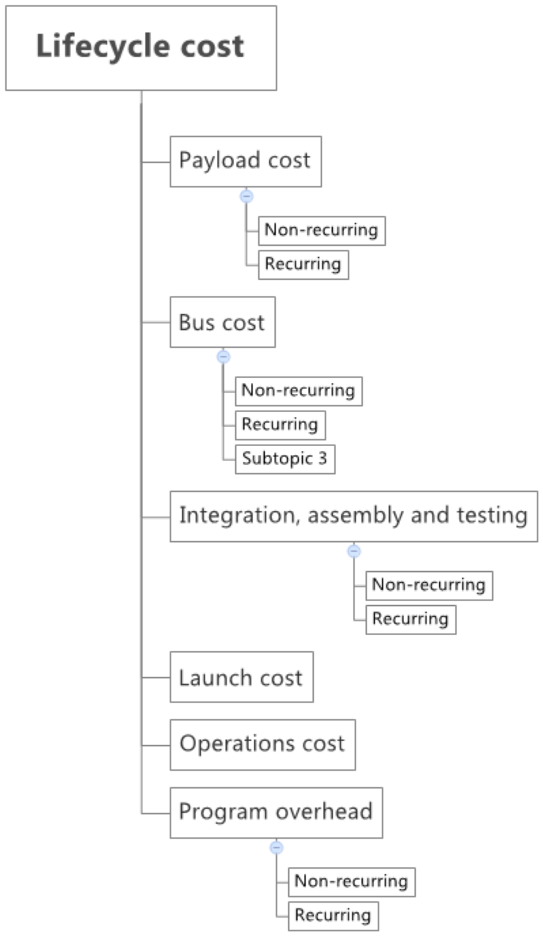

Figure 19. Lifecycle cost breakdown

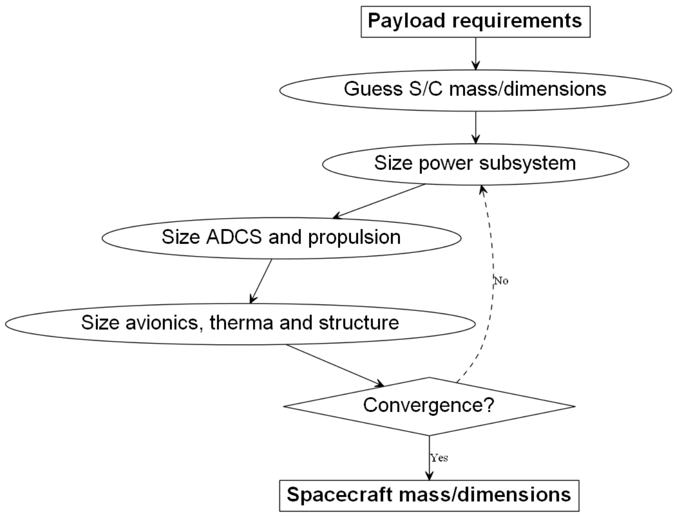
Payload requirements
Guess S/C mass/dimensions
Size power subsystem
Size ADCS and propulsion
Size avionics, therma and structure
Convergence?
No
Yes
Spacecraft mass/dimensions

Figure 20. Spacecraft design algorithm

Acknowledgments

The authors would like to thank la Caixa Fellowship program, NASA, and Draper Laboratory for partially funding this project.

References

[1] E. Crawley, O. D. Weck, S. Eppinger, C. Magee, J. Moses, W. Seering, J. Schindall, D. Wallace, and D. Whitney, "Engineering Systems Monograph," Massachusetts Institute of Technology, Tech. Rep., 2004.

[2] N. P. Suh, "Complexity Theory Based on Axiomatic Design," in *Complexity: Theory and Applications*. New York: Oxford University Press, 2005, ch. 3, pp. 54–86.

[3] M. W. Maier, "Architecting principles for systems-of-systems," *Systems Engineering*, vol. 1, no. 4, pp. 267–284, 1998.

[4] M. W. Maier and E. Rechtin, *The Art of Systems Architecting*. New York: CRC press, 2000.

[5] T. Weilkiens, *Systems engineering with SysML/UML: modeling, analysis, design*. Heidelberg, Germany: The Morgan Kaufmann/OMG Press, 2006.

[6] DoD Architecture Framework Group, "DoD Architecture Framework version 2.1," Washington DC, 2000.

[7] M. Rao, S. Ramakrishnan, and C. Dagli, "Modeling and Simulation of Net Centric System of Systems Using Systems Modeling Language and Colored Petri-nets : A Demonstration Using the Global Earth Observation System of Systems," *Systems Engineering*, vol. 11, no. 3, pp. 203–220, 2008.

[8] D. Dori, I. Reinhartz-Berger, and A. Sturm, "OPCAT-a bimodal CASE tool for object-process based system development," in *5th International Conference on Enterprise Information Systems (ICEIS 2003)*, 2003, pp. 286–291.

[9] N. Olsen, R. Haagmans, T. J. Sabaka, A. Kuvshinov, S. Maus, M. E. Purucker, M. Rother, V. Lesur, and M. Mandea, "The Swarm End-to-End mission simulator study : A demonstration of separating the various contributions to Earths magnetic field using synthetic data," *Earth, Planets, and Space*, vol. 58, pp. 359–370, 2006.

[10] R. M. Atlas, "Observing System Simulation Experiments: methodology, examples and limitations," in *Proceedings of the WMO Workshop on the Impact of various observing systems on Numerical Weather Prediction*, Geneva, Switzerland, 1997.

[11] M. Adler, R. C. Moeller, C. S. Borden, W. D. Smythe, R. F. Shotwell, B. F. Cole, T. R. Spilker, N. J. Strange, A. E. Petropoulos, D. Chattopadhyay, J. Ervin, E. Deems, P. Tsou, and J. Spencer, "Rapid Mission Architecture Trade Study of Enceladus Mission Concepts," in *Proceedings of the 2011 IEEE Aerospace Conference*, Big Sky, Montana, 2011.

[12] J. Hauser and D. Clausing, "The house of quality," *Harvard Business Review*, no. May-June 1998, 1988.

[13] T. L. Saaty, "Decision Making With the Analytic Hierarchy Process," *International Journal of Services Sciences*, vol. 1, no. 1, pp. 83–98, 2008.

[14] A. M. Ross, D. E. Hastings, J. M. Warmkessel, and N. P. Diller, "Multi-attribute Tradespace Exploration as Front End for Effective Space System Design," *Journal of Spacecraft and Rockets*, vol. 41, no. 1, 2004.

[15] W. L. Baumol, "On the social rate of discount," *The American Economic Review*, vol. 58, no. 4, pp. 788–802, 1968.

[16] M. K. Macauley, "The value of information : Measuring the contribution of space-derived earth science data to resource management," *Journal of Environmental Economics and Management*, vol. 22, pp. 274–282, 2006.

[17] S. Jamieson, "Likert scales: how to (ab)use them." *Medical education*, vol. 38, no. 12, pp. 1217–8, Dec. 2004.

[18] R. C. Mitchell and R. T. Carson, *Using Surveys to Value Public Goods: The Contingent Valuation Method*, S. Aller, Ed. Washington DC: Library of Congress, 1989.

[19] O. C. Brown, P. Eremenko, and P. D. Collopy, "Value-Centric Design Methodologies for Fractionated Spacecraft: Progress Summary from Phase 1 of the DARPA System F6 Program," *AIAA SPACE 2009 Conference & Exposition*, 2009.

[20] a. Stoffelen, G. J. Marseille, F. Bouttier, D. Vasiljevic, S. de Haan, and C. Cardinali, "ADM-Aeolus Doppler wind lidar Observing System Simulation Experiment," *Quarterly Journal of the Royal Meteorological Society*, vol. 132, no. 619, pp. 1927–1947, Jul. 2006.

[21] A. Newell and H. A. Simon, *Human Problem Solving*. Englewood Cliffs, NJ: Prentice Hall, 1972.

[22] R. Lindsay, B. G. Buchanan, and E. A. Feigenbaum, "DENDRAL: A Case Study of the First Expert System for Scientific Hypothesis Formation," *Artificial Intelligence*, vol. 61, no. 2, pp. 209–261, Jun. 1993.

[23] B. G. Buchanan and E. H. Shortliffe, *Rule-based Expert Systems: the MYCIN experiments of the Stanford Heuristic Programming Project*. Addison-Wesley, 1984.

[24] P. Hart, R. Duda, and M. Einaudi, "PROSPECTORA computer-based consultation system for mineral exploration," *Mathematical Geology*, no. November 1977, 1978.

[25] J. McDermott, "R1: A Rule-Based Configurer of Computer Systems," *Artificial lntell., 19: 39*, vol. 19, no. 1, pp. 39–88, Sep. 1982.

[26] K. J. Healey, "Artificial Intelligence Research and Applications at the NASA Johnson Space Center," *AI Magazine*, vol. 7, no. 3, pp. 146–152, 1986.

[27] C. Forgy, "Rete: A fast algorithm for the many pattern/many object pattern match problem," *Artificial intelligence*, vol. 19, no. 3597, pp. 17–37, 1982.

[28] L. A. Zadeh, "Fuzzy Sets," *Information and Control*, vol. 8, no. 3, pp. 338–353, Jan. 1965.

[29] C. Haskins, "INCOSE Systems engineering handbook - A guide for system life cycle processes and activities," *Systems Engineering*, 2006.

[30] N. Das, D. Entekhabi, and E. Njoku, "An Algorithm for Merging SMAP Radiometer and Radar Data for High-Resolution Soil-Moisture Retrieval," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 49, no. 99, pp. 1–9, 2011.

[31] A. Messac and A. Ismail-Yahaya, "Multiobjective robust design using physical programming," *Structural and Multidisciplinary Optimization*, vol. 23, no. 5, pp. 357–371, Jun. 2002.

[32] B. Cameron, “Value flow mapping: Using networks to inform stakeholder analysis,” *Acta Astronautica*, vol. 62, no. 4-5, pp. 324–333, Feb. 2008.

[33] R. Yager, “On ordered weighted averaging aggregation operators in multicriteria decision making,” *Systems , Man and Cybernetics, IEEE Transactions on*, no. 1, pp. 183–190, 1988.

[34] J. Fortin, D. Dubois, and H. Fargier, “Gradual Numbers and Their Application to Fuzzy Interval Analysis,” *IEEE Transactions on Fuzzy Systems*, vol. 16, no. 2, pp. 388–402, Apr. 2008.

[35] H. Apgar, “Cost Estimating,” in *Space Mission Engineering: The new SMAD*. Hawthorne, CA: Microcosm, 2011, ch. 11.

[36] Chalmers University of Technology, “Use of P-band SAR for forest biomass and soil moisture retrieval,” European Space Agency, Tech. Rep., 2004.

[37] D. Selva, “Rule-based system architecting of Earth observation satellite systems,” PhD dissertation, Massachusetts Institute of Technology, 2012.

[38] H. H. Agahi, G. Ball, and G. Fox, “NICM Schedule & Cost Rules of Thumb,” in *AIAA Space Conference 2009*, no. September, Pasadena, CA, 2009, pp. 6512–6512.

BIOGRAPHY

Daniel Selva *received a PhD in Space Systems from MIT in 2012 and he is currently a post-doctoral associate in the department of Aeronautics and Astronautics at MIT. His research interests focus on the application of multidisciplinary optimization and artificial intelligence techniques to space systems engineering and architecture, in particular in the context of Earth observation missions. Prior to MIT, Daniel worked for four years in Kourou (French Guiana) as a member of the Ariane 5 Launch team. In particular, he worked as a specialist in operations concerning the guidance, navigation and control subsystem, and the avionics and ground systems. Daniel has a dual background in electrical engineering and aeronautical engineering, with degrees from Universitat Politecnica de Catalunya in Barcelona, Spain, and Supaero in Toulouse, France. He is a 2007 la Caixa fellow, and received the Nortel Networks prize for academic excellence in 2002.*

Edward F. Crawley *received an Sc.D. in Aerospace Structures from MIT in 1981. His early research interests centered on structural dynamics, aeroelasticity, and the development of actively controlled and intelligent structures. Recently, Dr. Crawley's research has focused on the domain of the architecture and design of complex systems. From 1996 to 2003 he served as the Department Head of Aeronautics and Astronautics at MIT, leading the strategic realignment of the department. Dr. Crawley is a Fellow of the AIAA and the Royal Aeronautical Society (UK), and is a member of three national academies of engineering. He is the author of numerous journal publications in the AIAA Journal, the ASME Journal, the Journal of Composite Materials, and Acta Astronautica. He received the NASA Public Service Medal. Recently, Prof Crawley was one of the ten members of the presidential committee led by Norman Augustine to study the future of human spaceflight in the US.*