Engineering Systems Integration, Testing, and Validation

Ricardo Valerdi and Brendan P. Sullivan

Contents

Abstract

Developing and implementing interventions in engineering systems must undergo rigorous testing before being deployed into their operational

R. Valerdi (✉)
Department of Systems & Industrial Engineering, University of Arizona, Tucson, AZ, USA
e-mail: rvalerdi@arizona.edu

B. P. Sullivan
Department of Design, Production and Management, University of Twente, Enschede, Netherlands
e-mail: b.p.sullivan@utwente.nl

A. Maier et al. (eds.), *Handbook of Engineering Systems Design*,
https://doi.org/10.1007/978-3-030-46054-9_20-1

environment. An engineering system's complexity determines the sophistication of the testing needed to demonstrate its ability to meet intended objectives. In this chapter, we explore the various testing methodologies that shed light on the behaviour of a system. This section introduces the reader to the general role of life cycle cost of systems in testing, including reference engineering systems that can leverage such a testing approach. Within section "Engineering Systems Considerations and Interventions", the implications of various types of system complexity are presented along with the implications that tight and loose coupling can have on systems. This section includes an example taken from the International Space Station that illustrates various considerations involved with testing engineering systems. Section "Testing Methodologies" discusses the finer details of systems testing by presenting current challenges for testing engineering systems, suitable testing approaches, and the roll of test planning (design of experiments) in engineering systems development. Section "Developments in Testing" of this chapter presents an example of a drone delivery system that leverages a decision support system to help optimise test strategies in situations where the system is too complex to test manually, and trade-offs must be made between test coverage, cost, and delivery time. At the heart of the methodology presented in this section is an algorithm that can help lead to smarter testing decisions through the prioritisation and sequencing of tests. This is accomplished by integrating a parametric cost model, knowledge gradient algorithm, and Bayesian updating algorithm. The chapter aims to support systems engineers coordinate and plan tests that help decision-makers learn as much about a system as quickly as possible while gaining confidence that the system is ready for deployment.

Keywords

Complexity · Coupling · Engineering systems · Grand challenges · Life cycle · Testing · Validation · Verification

Introduction

Systems design is the organised and structured application of processes that result in the development, production, deployment, training, operation and maintenance, refinement, and retirement of a system (Rasmussen 2003). The end goal is to develop systems that deliver value for stakeholders by fulfilling requirements, ensuring effective interfaces, and validating specific system objects on time and cost. The engineering of systems provides and allows for both creative design alternatives for meeting system objectives on paper and technical competence to ensure these objectives are also delivered in real life. This is achieved through the design of components, configuration items and integration, across the entire life cycle of the system. Systems can be described according to their purpose/objectives, complexity, social dimension, and technical elements, namely:

- **System**: is a construct or collection of different elements (interacting components) that together produce results not obtainable by the elements alone (Blanchard 2004).
- **System Task/Function**: An action, a task, or an activity performed to achieve a desired outcome (Hitchins 2007).
- **Complex Sociotechnical Systems:** are complex purpose-built systems composed of numerous interconnections, interactions, or interdependencies between social, managerial, and technical elements that are difficult to describe, understand, predict, manage, design, and/or change (Cherns 1976).

When designing engineering systems, it is necessary to consider the technical and social complexities brought upon by the needs being met and the critical functions required to meet those needs. To support this, the chapter focuses on the role, methods, and value of testing. We consider various testing approaches and techniques that help bring engineering systems to life. Given the current challenges in testing complex engineering systems, we highlight an approach that considers uncertainty, value-based decision-making, and cost that jointly can help bring increase our understanding of system behaviour in a short amount of time.

Sociotechnical Systems Theory

According to Cherns (1976), sociotechnical systems are systems in which both human and non-human elements interact to deliver societal value in some way. This is one of many possible ways to describe engineering systems. In order for any engineering systems to be achieved, societal (people) and technical aspects must be considered and effectively organised to improve how we interact, interconnect, and collaborate.

Sociotechnical systems theory has many connotations and applications ranging from engineering to management and education, the commonality between all disciplines being the interaction between social and technical factors that characterise the successfulness of system development (Emery and Trist 1960; Trist 1981; Baxter and Sommerville 2011). Introduced in Section "Sociotechnical Systems Theory", the grand challenges facing the world require an integrative holistic view of large-scale, complex, technologically enabled systems designed with a sociotechnical perspective.

Examples of sociotechnical systems can range from energy distribution systems that distribute mixed energy (solar, wind, hydro, and non-renewable) across a grid network to an intermodal transportation system where rail, commercial vehicles, maritime, or air transportation come together (see Fig. 1) to provide a valuable service for society. In developing and considering such sociotechnical systems, it is

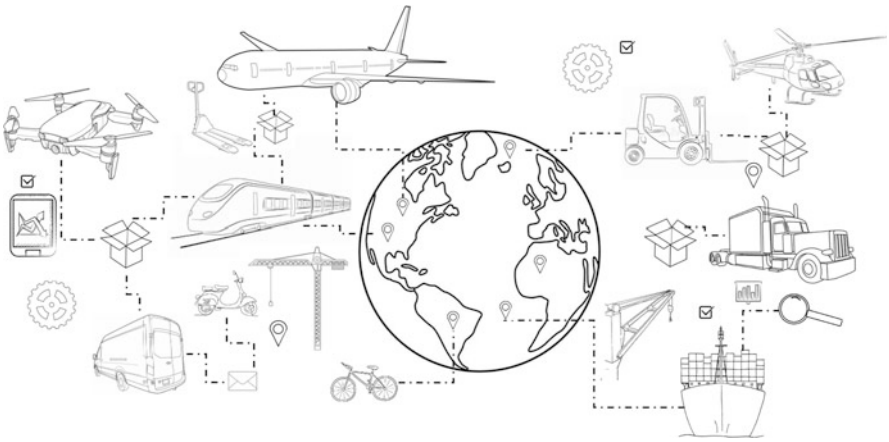

Fig. 1 Intermodal system – improving infrastructure

essential to realise that decisions should be made to set up the possibility for future innovation and forward-thinking solutions, which can be done by carefully considering and evaluating the implications of different decision scenarios on the system across multiple points of time.

Beyond Cherns, additional aspects of sociotechnical systems are proposed by Clegg (2000):

1. The interaction between social and technical factors supports the successful (or unsuccessful) performance of a system.
2. Purposeful and goal-directed functions that deliver value to society in some way.
3. Interrelated, with strong dependencies that allow each aspect to complement and benefit the other enabling collective improvement and optimisation.

According to Cherns (1976) and Clegg (2000), sociotechnical systems utilise multifunctional, multilevel, multidisciplinary teams to develop systems that are capable of delivering societal value in an interconnected and collaborative manner. By considering:

- Compatibility – Process/function is compatible with objectives.
- Minimal critical specification – Specify no more than what is necessary, while always specifying what is essential.
- Variance within the system – Any unplanned/unprogrammed event, which creates deviation in quality, responsiveness, and function.
- Multifunctional parts/components – Institution of choices that can perform functions or meet the systems objectives by using combinations of elements.

(continued)

- Boundaries – Multidimensional clusters that operate to support the distribution of work rather than consolidated groupings.
- Information flow – Deliverance and generation of information/data where it is needed, and in the format desired and according to the necessary and specified context (correct information, for the correct person/system, at the correct time).
- Support congruence – The system should support and reinforce social principles.
- Design and human values – Provide quality system solutions without limiting or suppressing the values of those the system was built to benefit.
- Inherent incompleteness of the system – Recognition that upon completion and deployment of the system, its consequences will necessitate redesign.

The design of sociotechnical systems considers the aspects previously discussed while also considering the objectives that the overall system is trying to achieve. This requires that the social and technical elements provide for and support the innovativeness of the people involved to identify and establish goals that can be attained through interrelated optimisation. This includes, but is not limited to, how machines and technical systems behave and how people interact with them to bring external forces (political, ecological, and societal) into the design process. This interaction between each of these forces impacts the inherent complexity of the system by considering compatibility, minimal critical specification, variance within the system, multifunctional parts/components, boundaries, information flow, support congruence, design and human values, as well as the inherent incompleteness of the system. For example, it may not be as simple as testing to make sure a system meets the end user's needs. The impact of the system on its owners, operators, government regulators, taxpayers, and competitors may very well be as crucial as the agreed-upon technical requirements between client and developer. Consider the social and political implications of electric vehicles as an example of the broad and far-reaching implications that extend beyond the manufacturer and driver relationship. This highlights the complexities involved in testing and evaluating the performance of engineering systems during their design phase, instead of simply observing (and most likely criticising) it during its operational life.

US National Academy of Engineering Grand Challenges

There are many types of engineering systems developed to overcome challenges on a global and individual level. In considering design, engineering systems are generally forward-thinking solutions that solve problems related to humanity, ranging from the exploration of space and planetary science, sustainability, health, security, and general human need. The purpose for engineering systems in each of the facets of

Table 1 US National Academy of Engineering Grand Challenges (2021)

Challenge	Explorative solutions and systems
Making solar energy more economical	Development of future generation hybrid solar cells with organic semiconductors and inorganic nanostructures Solar-powered aircraft – Solar Impulse (2016), Airbus Zephyr S HAPS (2018), Boeing Odysseus (2019)
Provide energy from fusion	Joint European torus (JET) and the mega amp spherical tokamak (MAST) in the United Kingdom International Thermonuclear Experimental Reactor (ITER), currently under construction in Cadarache, France (2020)
Carbon sequestration methods	Sleipner A project Climeworks direct carbon capture plant (Switzerland)
Manage the nitrogen cycle	Solar glass Smart fertilisers
Provide access to clean water	Desalination Water reclamation Smart irrigation
Restore and improve urban infrastructure	Intermodal transportation systems Smart grids
Advance health informatics	Remote patient monitoring Electronic medical records Master patient index
Engineer better medicines	Rapid diagnostic systems Personalised medicine (theranostics)
Reverse engineer the brain	Artificial intelligence Neural prostheses
Prevent nuclear terror	Passive nuclear material monitoring Nuclear screening systems (e.g. nuclear car wash)
Secure cyberspace	Self-healing computer systems Cyber-attack-resilient architecture for next-generation electricity distribution systems
Enhance virtual reality	Augmented and virtual cognitive systems
Advance personalised learning	Evolutionary educational presentation systems Educational recommender systems
Engineer the tools of scientific discovery	NASA space launch system OSIRIS-REx Mission

life differ, though irrespective of their addressed facet, a deep understanding of the problem being addressed improves deciding which functions are better and what the is best way to test such functions. Table 1 provides an overview of systems and explorative solutions that have been developed or are under development according to the United State National Academy of Engineering Grand Challenges (2021).

For the systems and solutions introduced in Table 1 to be successfully developed, accompanying testing strategies must be considered early in the life cycle. For testing to inform decision-making, the testability of such systems has to be treated with the same importance as other attributes like reliability, interoperability, sustainability, and survivability. Testing considerations grow in importance as systems grow

in complexity because the more uncertainty there is about a system's performance, the more we must invest in ensuring that it is ready for deployment (DoD 2004; Potts et al. 2020). Test results are also crucial in the early-stage exploration of the problem and solution space, giving decision-makers a chance to (re-)evaluate the feasibility and associated costs of alternative solutions.

The Role of Life Cycle Cost of Systems in Testing

Every human-designed engineering system has a life cycle and cost. The life cycle cost (LCC) of an engineering system is the total cost over its entire life span, including development, verification, testing, validation, and disposal (Mooz et al. 2003). Despite every engineering system (product or solution) having a cost, there historically has been an emphasis by engineers to focus on the performance of a system with less regard for the downstream costs.

> LCC is the total cost of all costs related to or associated with a system from cradle to grave (development, verification, testing, validation, and disposal) and explicitly accounts for the time value of money (the variation in the cost of an expenditure relative to its timing).

LCC is used to quantify the costs associated with an engineering system throughout each phase of its anticipated life. It supports the selection of economically viable and innovative solutions while ensuring that all aspects of the system integrate and function according to the requirements and needs of the stakeholders. Through this economic approach, LCC supports decision-makers to identify and choose the cost-effective approach from a series of alternatives to deliver the greatest value to the stakeholder at the lowest long-term cost (Farr 2012).

LCC can be a valuable tool used for systems decision-making (passing between decision gates). As shown below, the general system life cycle consists of five stages: conceptualisation, design, development, production and testing, and operation and retirement of the system (disposal). Each of the life cycle phases has costs and must be carefully considered since the consequences of making early decisions without accurate analysis can dramatically impact the percentage of system costs against time required for development. Considering Fig. 2, we can infer that by the production and testing phase of a new system, 50% of the total LCC has been spent, while 95% of the funds have been committed based upon decisions made by the developer or client. The cost to extract defects represents the cost associated with fixing unanticipated problems. The later the defect occurs, the more costly it becomes to rectify it.

LCC estimates are built upon the projections of total cost to the funding organisation for the system ownership and acquisition over its entire life span. This may include the costs of the direct, indirect, recurring, non-recurring, and other related

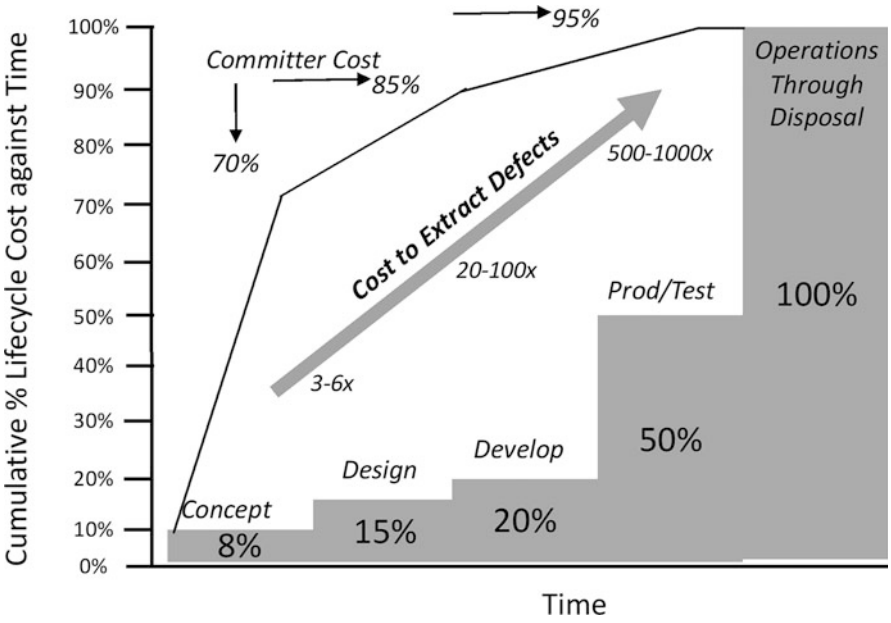

Fig. 2 Committed life cycle cost against time (INCOSE 2015)

costs incurred or estimated to be incurred during the design, such as research and development, investment, operations, maintenance, support, disposal, and other relevant costs. It is essential to consider the phases of LCC, because the upfront acquisition phase is only a small part in relation to the total cost. Yet it plays a significant role in the decisions made that will have downstream effects (DOD 1983; Farr 2010; Kerzner 2017).

Testing therefore also has to include testing the validity of life cycle cost estimates, as well as generate the necessary knowledge to improve the accuracy of life cycle cost estimates.

Engineering Systems Considerations and Interventions

When designing engineering systems, it is necessary to consider the technical and social complexities brought upon by the needs being met and the critical functions required to meet those needs. When referring to large-scale, sociotechnical engineering systems design, these systems are always partially designed and partially evolved (de Weck et al. 2011). Practically, that means we will primarily look at interventions in existing systems rather than a complete redesign. This presents an even more complicated scenario because not all decisions can be made in the interest of optimal systems design. Some decisions, large and small, are already hard-wired

due to constraints that result from previously existing infrastructure, culture, or processes.

A system is only as good as the tests imposed to it, and a test can be good only if you can clearly understand the numerous interconnections, interactions, or interdependencies that you will be analysing. To do this, we must consider the trade-offs between requirements, functions, and alternate system resources that have/will take place to achieve a valuable, cost-effective, life cycle balanced system that maximises stakeholder desires (Blanchard et al. 1990). The following best practices synthesise several considerations that support the eventual goal of delivering a successful system (Bartolomei et al. 2011).

- Clearly articulated problem definition, stakeholders (and relationships), system mission, and environment
- Definition of the problem including context and external systems, the system must interact or interface with
- Balancing of needs between stakeholders and engineering, throughout development (communication), including the articulation of system needs, and translation of requirements
- Matching trade-offs, to make requirements more transparent and support the identification of requirements that are not feasible
- Articulation of critical functions and relationships to the user (human-robot interaction, cost, complexity, risk to human life, etc.)
- Testing plan to mitigate risk before deployment

Complexity in Sociotechnical Systems

All systems have some inherent level of complexity, but the formation and magnitude of their complexity differ. It is essential to understand how complexity impacts design and by connection how it impacts its testing. It is therefore necessary to consider both technical and human complexity when working with an engineering system.

Contemporary technology-centric systems are diverse and contain higher degrees of systems complexity than ever before, requiring more knowledge than ever before to understand the operational functions and goals of the systems (Philbin 2008). Yet, the dimension of complexity may be overlooked, not fully understood, and often underestimated within systems design and development processes. Thus, as technologies advance and the magnitude of systems effort intensifies, systems complexities and uncertainties increase simultaneously (NASA 2007).

To focus the conversation, it is helpful to determine the types of complexities present in engineering systems and their influence on cost throughout the life cycle. McShea (1996) suggests that understanding the nature and state of complexity is a complex subject matter itself, regardless of the types or origins that a particular type of complexity may reside within a system. Recognising this challenge, Sheard (2013) identified four entities prevalent in complex engineering systems.

Complexity types (per Sheard 2013):

- **Project**-related complexity represents complexity types the organization developing the system. This is considered since the organization performing the work is generally already in existence and therefore has people already in place who work with others and is responsible for allocating responsibility for product realization tasks.
- **System**-related complexity refers to technological considerations and how the system is composed. This is the most commonly thought of complexity.
- **Environment**-related complexity refers to the 'Way Things Are' and can extend to include both external factors and stakeholders. Which identify/determine other systems that the system being developed must interface with as well as the technological environment.
- **Cognition**-related complexity emphasises the human aspect through the consideration of individual limitations and the actions in place to reduce risk and uncertainty.

Considering such complexity allows for the reduction and/or better management of potentially detremental impacts that can effect test efficiency and system success. As described in Section "Example: International Space Station" project complexities can have direct implications on the cost, overrun, and scheduling of development, system type complexities that reflect the *number of systems to be integrate and the number of interfaces that can increase the complexity of tests required to be performed (it is critical that test plans* minimise redundancy and maximise test efficiency). Moving beyond complexity, it is imperative to clearly define the boundaries of the system itself in order to determine how it will be tested.

System Boundaries

System boundaries are a fundamental part of engineering systems. The purpose of boundaries is to develop conceptual separation between the important elements of the system (relevant component) and its environment (external elements that can affect or be affected by the system). Engineering systems are bounded by component limits of control and are aligned with the system's purpose. Drawing the boundary correctly is crucial to systems design, development, and testing because to solve a systems problem, you must first know what the system is.

Context is understood as everything beyond the system's boundary. This includes the environment and the source for inputs and the later destination for system outputs. The context affects the general nature of the inputs and interpretation of the systems outputs, and stakeholders need to be cognisant of contexts that can affect

the system. Since the context and, by extension, the environment are outside of the system and cannot be controlled, it causes uncertainty for the developers.

In the case of an engineering system, the boundaries and internal interfaces can be documented in multiple ways, including through either interface control document or interface control specification. The importance of understanding interfaces concerning system boundaries is that stakeholders must understand the assemblage of the system, the functions, and capabilities for the development to be successful. Through this understanding, combined interactions, including processes and data flow, within and across the system facilitate the modelling and evaluation of system behaviour and performance to better understand and plan testing.

Useful lexicon for understanding boundaries in engineering systems

- Boundary: separates the system's internal elements and processes, from external factors or elements.
- Context: defines the development and operational space outside of the system boundary, illustrating the interaction between elements.
- Environment: exogenous factors or elements that affect or can be affected by the engineering system (internal factors and elements).
- System: is a construct or collection of different elements (interacting components) that together produce results not obtainable by the elements alone. Can be understood as a group of components that interact together and are necessary for fulfilling a purpose.
- Subsystem: is a system in its own right, except it normally will not provide a useful function on its own, it must be integrated with other subsystems (or systems) to make a system.
- Elements: are not restricted to hardware but can also include software and can even include people, facilities, policies, documents, and databases.
- Component: are elements that make up a subsystem or system and are dependent on other components (interact with each other) to create the system's behavior.

Demystifying Integration through Coupling

Another dimension that is important in understanding engineering systems complexity and testing is coupling (Marais et al. 2004). That is, how connected an engineering system's parts are to each other. The degree of coupling within an engineering system may be described across a range of tight coupling to loose coupling. Tight coupling is when components are highly dependent on one another, while loose coupling is when there is little or no dependency between components. As shown in Fig. 3, the differences between tight and loose coupling can also be described in terms of coordination and information flow.

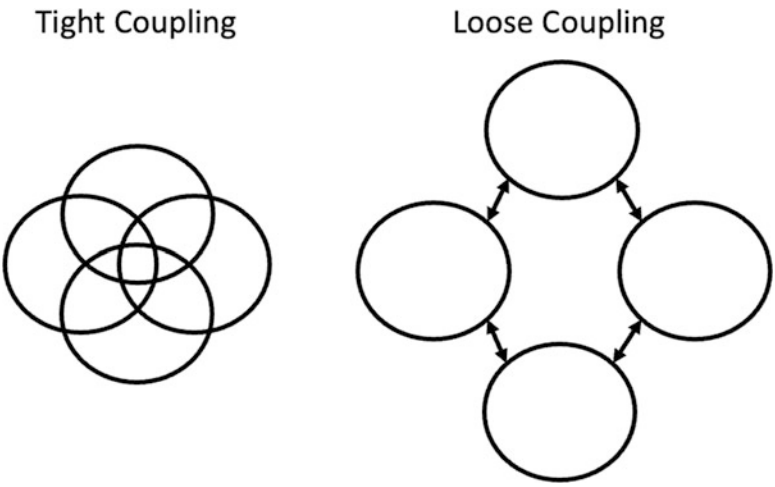

Fig. 3 Tight vs. Loose Coupling

The decision about the degree of coupling in an engineering system may be driven by architectural attributes like quality, security, flexibility, interoperability, reliability, performance, and many others (Elias and Jain 2017). For instance, to maintain high levels of security, an engineering system might be loosely coupled to create isolation between system components in the event one of them is breached. In some cases, the degree of coupling may not be negotiable because of legacy considerations. Many engineering systems have inherited traits that are unchangeable and therefore decrease the number of solutions that can be implemented to meet their objectives.

In either case, the degree of coupling significantly impacts the testing of engineering systems. In loosely coupled cases, more tests may be required to ensure the functionality is adequately working. In tightly coupled cases, however, higher connectivity may be an advantage because it might require less tests to observe the system's behaviour.

As always, there are exceptions to these examples. For instance, tightly coupled systems may require all of the components to be available and engaged in the test. This may not be a simple or inexpensive task. Similarly, loosely coupled systems may be simpler to test by undergoing testing at different times and locations, facilitating coordination and data collection.

Example: International Space Station

The International Space Station (ISS) is one of the largest and most complex engineering systems, developed out of a collaborative effort between the United States and Russia to provide for an on-orbit habitable laboratory for scientific and research activities (International-Space-Station-Program-Science-Forum

2015). Stockman et al. (2010) provide a detailed case study of the ISS from the systems engineering perspective. The complexity of the system can be seen throughout the development process taking form in the system, the environment, cognition, and more specifically in project (Section Complexity in Sociotechnical Systems). The project complexity was related to many elements, but none so significant as the inclusion six international partners that were tasked to collaborate in the building of 87 flight elements integrated over 44 assembly flights during a 5-year time frame. Although both the National Aeronautics and Space Administration (NASA) and the Roscosmos State Corporation for Space Activities had significant experience in multi-national complex space system development, the partnership and integration required to develop the ISS was overwhelming, leading to project delays, overruns, and integration issues (Thomas 1996).

Complexities within the ISS:

- **Project** – multiple space agencies involved in building the technical system; NASA (United States) and ROSCOSMOS (Russia)
- **Technical System** – Systems being designed; ISS Vehicle, ISS Flight Elements, ISS Launch Package
- **Environment** – the technological environment of ISS stakeholders, and the technological environment into which the system will be inserted when built.
- **Cognition** – Varying approaches to system engineering, design, and development (NASA and ROSCOSMOS)

As a sociotechnical system, the ISS can be further described through the multifunctional, multilevel, multidisciplinary international teams used to develop the engineering system and the critical societal value derived from the research and experiments performed. According to the International Space Station Program Science Forum (2015), the research performed on the ISS to benefit humanity includes the following:

- Development of health technology
- Solutions to prevent bone loss
- Human immune system defences
- Medical treatments and therapies
- Food and the environment
- Heart health and biorhythms
- Improving balance and movement

Many boundaries existed in the development; however, two observable and critical areas of emphasis delineating internal process from other system elements

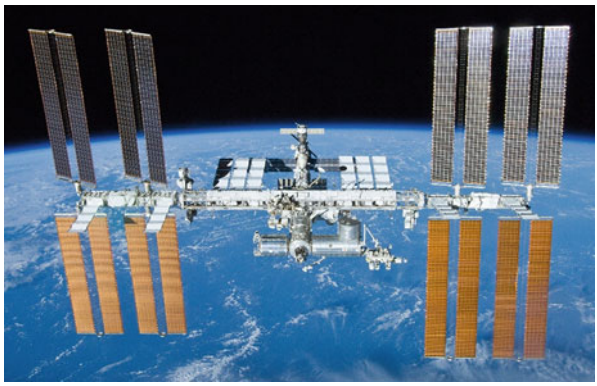

Fig. 4 International Space Station (NASA – public domain image)

(Section System Boundaries) were the (1) Vehicle Systems Engineering & Integration (SE&I), the SE&I of the of all the flight elements of ISS Vehicle with each other, and (2) Launch Package SE&I, the SE&I of the individual ISS Vehicle flight elements with the other constituents of an assembly mission, such as launch vehicle integration. The ISS was designed as a network of distributed subsystems that were interconnected among discrete elements (Section Demystifying Integration through Coupling). Each subsystem and element introduced unique design and integration challenges independently, though for the ISS to be physically interconnected and functionally interoperable, the distributed systems and discrete elements also needed to be integrated seamlessly into a unified space vehicle (Stockman et al. 2010). This integration concept made the planning and execution of testing for the ISS both difficult and complex since during building each flight element, the subsystem teams would develop their subsystems to meet the respective performance requirements during each stage of the ISS Vehicle assembly (see Fig. 4).

The integrated performance of all subsystems at each stage of the ISS Vehicle assembly mutually determined mission success (Stockman et al. 2010).

Testing during the development of the ISS presented a major challenge to the NASA and its partners Stockman et al. 2010. Despite the ISS Program verification philosophy to "integrate and test on the ground what we fly before we fly", new and innovative methods to test and verify interfaces were required. Reasons for this were attributed to many of the modules being developed in different countries and delivered "just in time" for the launch, leaving minimal time for integration testing. Therefore, before deployment, each module had to be tested for its own internal operation. Then it had to interface with the launch vehicle, and finally, it had to work in space while integrated with multiple modules and systems (Stockman et al. 2010). Even if an individual module performed well by itself, the success of the ISS depended on the ability of all modules to perform well as an integrated whole.

We now shift our attention from describing the complexities of engineering systems to exploring specific methodologies that can aid in successful testing.

Testing Methodologies

Before discussing specific methods and their application domain, it is beneficial to begin by defining commonly used terminology in the testing context, particularly when working in a multifunctional, multilevel, or multidisciplinary team. This common vernacular ensures conformity throughout the testing process and should be applied to all testing methods. It is likely impossible to test for every imaginable scenario, due to complexity or cost (Barhorst et al. 2011). The objective of the test should therefore be well defined and correspond to the capabilities of the selected method. Vague or open-ended verification and validation plans lead to project overruns, system failure, and loss of confidence from the stakeholders (Wheatcraft 2012).

The terms *validation* and *verification* are sometimes confused and frequently mentioned in the incorrect order (e.g. verification and validation) (Ryan and Wheatcraft 2012). On the one hand, validation, which should be listed first, is the process of evaluating the final product to check whether it meets the customer expectations and requirements. In other words, it is the process of checking "Did I build the right system?"

On the other hand, verification is the process of testing documents, design, and functionality. It includes activities such as inspection (measurement to verify that the item conforms to its specified requirements), analysis (the use of established technical or mathematical models or simulations, algorithms, or other scientific principles and procedures to provide evidence that the item meets its stated requirements), and demonstration (actual operation of an item to provide evidence that it accomplishes the required functions under specific scenarios). In other words, it is the process of checking "Did I build the system right?". The order in which these are done is important because verification might be successful, but for a system that is the wrong thing for the client (it was not validated). That is why validation needs to occur before verification.

Challenges for Testing Engineering Systems

Engineering systems offer a unique opportunity for optimising testing because of their distributed nature, emergent properties, and dynamic topologies and boundaries. A key challenge is that additional capabilities can be gradually inserted in a system or environment and each insertion requires extensive testing before being deployed successfully in operational environments. This challenge can be addressed through systematic planning that leverages techniques from the "science of testing" that helps identify nearly optimal solutions that minimise cost while maximising test coverage (Young 2011). However, such a task is nontrivial because of the unstructured and dynamic environment of engineering systems. To address this challenge, we propose an innovative testing approach that blends techniques such as parametric cost modelling, knowledge gradient algorithms, and Bayesian updating algorithms.

These analytical techniques can help manage the delicate balance between performance, resiliency, and security level (Yang et al. 2012). The trade space

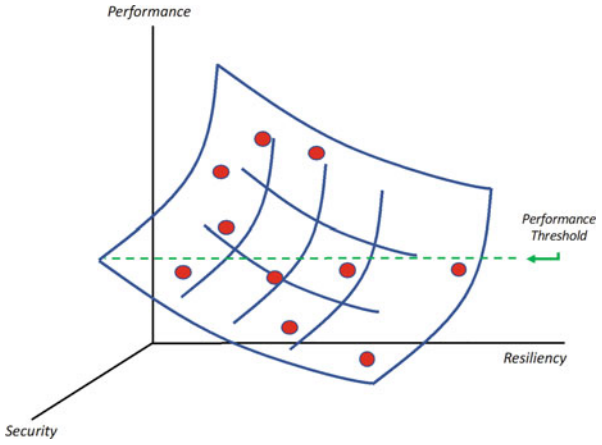

Fig. 5 Tradespace for engineering systems

Table 2 Different types of testing

Functional testing types	Non-functional testing types
Unit testing	Performance testing
Integration testing	Load testing
System testing	Stress testing
Sanity testing	Volume testing
Smoke testing	Security testing
Interface testing	Compatibility testing
Regression testing	Install testing
Beta/acceptance testing	Recovery testing
	Reliability testing
	Usability testing
	Compliance testing
	Localisation testing

illustration in Fig. 5 involves the selection of feasible solutions (dots) that meet or exceed the desired performance threshold (dashed line).

Even when feasible solutions are identified (dots), external considerations – such as government policy or political opposition – may still eliminate them from consideration. This is why the sociotechnical system approach may be helpful in comparing solutions from a broader, more holistic perspective.

Testing Approaches

Together, validation and verification are part of the broader concept of testing. There are many types of testing approaches, each with their advantages and disadvantages. The most common testing types, in Table 2, are categorised as functional and non-functional testing according to system performance type being verified.

Functional testing ignores the internal parts and focuses on the output to check whether intended requirements are met. It is a black box-type testing geared to the

functional requirements of a system (Myers 2011; Meinke 2004). Non-functional testing involves testing the "how" (non-functional requirements) a system will accomplish something and can be accomplished by implementing unique tests which are presented in Table 2, including load testing, stress testing, security, volume, recovery testing, etc. (Hooda and Chhillar 2015). The objective is to ensure whether the response time of a product is quick enough to meet the requirements.

Beyond the testing types described above, other broader approaches should be noted. Alpha testing aims to identify all possible issues or defects before releasing it into the market or to the user. Beta testing is a formal type of testing that the developer or the customer may carry out. It is performed in the real environment before releasing the product to the market for the actual end users.

Happy path testing aims to test an application successfully on a positive flow (Cohen et al. 2005). It does not look for negative or error conditions. The focus is only on the valid and positive inputs through which application generates the expected output. Negative testing employs the mindset of "attitude to break". It involves using incorrect data or invalid data or input. It validates that if the system provides an error or invalid input, it will behave as expected.

In risk-based testing, the functionalities or requirements are tested based on their priority (Amland 2000; Felderer and Schieferdecker 2014). Risk-based testing includes testing of highly critical functionality, which has the highest impact on business and in which the probability of failure is very high.

Exploratory testing is informal testing performed by the testing team. The objective is to explore the functionality and find existing defects (Itkonen and Rautiainen 2005; Itkonen et al. 2009). Sometimes it may happen that during this testing, a major defect is discovered that causes a system failure.

Acceptance testing is performed jointly between developer and client to verify whether the end to end flow of the system is as per the business requirements or not, and if it is as per the needs of the end user (Davis and Venkatesh 2004). This may involve both functional and non-functional testing. The client accepts the product only when all the features and functionalities work as expected.

Regardless of which test methodology is employed, there are a variety of measures of effectiveness to evaluate their usefulness. These include speed, fidelity, knowledge obtained, coverage, accuracy, risk reduction, and cost savings. Ultimately, testing should increase confidence that helps stakeholders decide whether a system is ready to be deployed into its operational environment.

Design of Experiments

A recent trend in testing has emphasised statistical methods for more efficient use of resources (Cohen et al. 1998). This is motivated by decreased testing budgets, more complex systems, more software-intensive systems, more upgrades to existing systems (i.e. evolutionary procurement), and greater interest in system reliability, availability, and maintainability (McQueary et al. 2009). One particular approach that has been favoured among the test community is design of experiments (Seglie

2010) which has a long tradition in product development (Coleman and Montgomery 1993; Montgomery 2004) and dates back to the eighteenth century.

> Design of experiments (DOE) is rooted in the ability to provide a cost-effective way to perform more rigorous test planning.

The design of experiments (DOE) methodology is rooted in providing a cost-effective way to perform more rigorous test planning. Its main benefit is identifying the real operational envelope of a system and identifying an efficient test design that covers that envelope. This is accomplished by using modern statistical software to predict the performance of a system based on its design factors and their interactions. The additional rigour provided by DOE results in higher confidence (low probability of accepting a flawed system), higher statistical power (low probability of rejecting a sound system), and breadth (knowledge across the operational spectrum).

Applying DOE to developmental testing – where testers can perform controlled experiments – is adequate. However, operational testing – where emergent behaviours are more likely to occur – DOE has significant limitations. These include the following:

1. The assumption that the entire trade space is known
2. The ability to automatically replan the test strategy as emergent behaviours appear
3. The assumption that the value of each test and the feature it is designed to test are constant
4. The assumption that the cost of each test is the same

Cohen, Rolph, and Steffey (1998, p. 3) state: "...effective use of statistical methods is not limited to a determination of the appropriate sample size so that a test yields interval estimates at the required level of statistical confidence. It would often be preferable, given a fixed test budget, to design a test aimed at maximising the amount of information from the resulting test data in order to reach supportable statements about system performance".

We propose an approach that addresses the limitations of DOE and provides an answer to the need to maximise information for the least amount of cost. To illustrate such an approach, we describe how it could apply to the case of an unmanned and autonomous system.

The Curse of Dimensionality

The benefits of engineering systems, in particular when tight coupling exists, introduce tremendous challenges for testing (Newman 2001). For instance, Metcalfe's law (Gilder 1993) states that the value of a telecommunications network

is proportional to the square of the number of connected users of the system (n^2). Following the same logic, more extensive networks are more expensive to test since the number of connections grows exponentially. This is known as the "curse of dimensionality" which describes the problem caused by the exponential increase in volume associated with adding extra dimensions to a mathematical space.

> The "curse of dimensionality" is the exponential increase in volume associated with adding extra dimensions to a mathematical space.

Solving multidimensional problems requires statistical techniques like Markov chains, Monte Carlo analysis, machine learning, Bayesian statistics, orthogonal arrays, and optimisation (Powell 2010). Researchers have explored the need for more testing of IT systems (Graves 2010) and advocated operational realism in testing (Stephens et al. 2008). However, neither Graves nor Stephens provides quantifiable recommendations for reducing the cost or schedule of testing. More recent work by Gibson (2012) showed that virtual machines can reduce testing time of engineering change orders by 11% by reducing setup and configuration time of Windows and Linux machines. While useful, the Gibson study did not explicitly address the curse of dimensionality problem associated with testing engineering systems.

The most rigorous study to date focused on reducing the cost of testing was done by Pfeiffer et al. (2011). Their results showed that test coverage of software systems is dramatically affected by test section strategy. However, the objectives of the Pfeiffer et al. study fall short in the following ways:

1. Their approach does not account for uncertainty in the information obtained by each test. What is missing is the estimate of a standard deviation to the value obtained from each test: the results of such tests are unknown during the planning process.
2. Similar to design of experiments, their approach assumes that the entire trade space is known in advance. What is missing is an approach that recognises that emergent behaviours will influence how the test strategy needs to evolve over time.
3. Similar to design of experiments, their approach assumes the cost of each test is the same. What is missing is an approach that uses a cost model to identify the approximate cost of each test independent of its perceived value.

With their increasing complexity, engineering systems fall into the class of systems where the science of testing can provide a step function improvement in the way they are tested. Accordingly, we propose an approach that addresses the limitations of DOE and the Pfeiffer et al. (2011) approach with the eventual goal of maximising information and minimising cost.

Developments in Testing

Extended life and increasing complexity dictates that testing methods and tools keep up with technological (different maturity levels), social/regulatory, and environmental requirements imposed on systems and systems of systems (SoS). These planned and potentially unplanned/undesired complications need to be considered when planning and performing testing. In respect to testing complex systems, approaches can address component-based technologies, design patterns, and resource allocation techniques. As discussed in this section, solutions to improve how tests are performed and planned through the utilisation of decision support systems (DSS), which support organisational decision-making are presented along with corresponding practical applications.

Application Example of Testing Methodology to an Unmanned and Autonomous System

As an example of an engineering system, an unmanned and autonomous system (UAS) like the DHL "Paketcopter" or Amazon's Prime Air delivery drone (Fig. 6), which provides package delivery, offers a unique opportunity for gradual technology insertion of automation due to task repetitiveness, relatively moderate sensory requirements, and the limited human exposure to safety risks. A key challenge is that the functional and non-functional elements of the system require extensive testing before they are deployed safely and effectively in operational environments. This challenge can be addressed through systematic test planning of heterogeneous, multi-agent autonomous systems. However, such a task is nontrivial because of the unstructured and dynamic environment of UAS operations. To address this challenge, an example for a test planning tool could incorporate a supervisory controller of the distributed agent-based platforms, a mission planner for human-robot tasking, and a decision support system (DSS) for extensive test planning validation.

Fig. 6 Delivery drone (Mollyrose89 – Own work, CC BY-SA 4.0)

This section shows an example of an innovative approach for testing engineering systems with emergent behaviour through the use of a decision support system and associated local linear or backstepping control algorithms (Madani and Benallegue 2008). The objective is to test the UAS control algorithms iteratively by exposing inner workings of heterogeneous agents, their interactions with the supervisory control system, and finally, the highest level of decision-making, mission planning system. To accomplish this, it is possible to integrate a parametric cost model, knowledge gradient algorithm, and Bayesian updating algorithm embedded in a DSS.

The integration of automated technology in UAS faces significant challenges. UAS operations can be highly unstructured, dynamically changing, and heterogeneous. The execution of basic tasks requires the use of multiple pieces of equipment in a coordinated manner. An additional difficulty emerges from the use of different UAS platforms developed by a diverse number of companies. These UAS operation features suggest the need for automation to enable the capability to work cooperatively and self-adapt to dynamic changes.

Novel ideas to systematically handle the challenges in the automation of UAS emerge from the fields of robotics and automation, mainly due to recent efforts on multi-agent robotics and control. Multi-agent systems consist of interconnected dynamical systems capturing the behaviour of the individual entities intertwined within each other. In multiple UAS operations, each UAS may be defined by an agent, whereas cooperative algorithms within a communication infrastructure may define the interactions within the multi-agent, networked, and heterogeneous system.

It is necessary to collect data and information from all agents, the supervisory system, and the mission planning system to address this need. Using such data, measure of effectiveness can be used to determine the maturity of the UAS and its control algorithms. By integrating DSS to the UAS control system, we can identify an efficient test strategy of a complex heterogeneous UAS. The benefits include the following:

1. Reduction in testing time, effort, and cost
2. Reduction in uncertainty, unpredictability, and risk in testing
3. Rapid identification of new UAS capabilities

Decision Support Systems for Test Planning

The validation of local control algorithms is at the heart of autonomous functionality and can be accelerated with the help of a dedicated DSS that provides optimal test strategies to be executed (Ferreira et al. 2010). This iterative planning-replanning cycle can help ensure that the most critical scenarios are tested first so that the limits of the various control algorithms can be determined in the shortest amount of time. The importance of tests can be determined based on multiple criteria such as criticality to the user, human-robot interaction, cost, complexity, risk to human life, etc. These criteria can be selected with the help of parametric cost models,

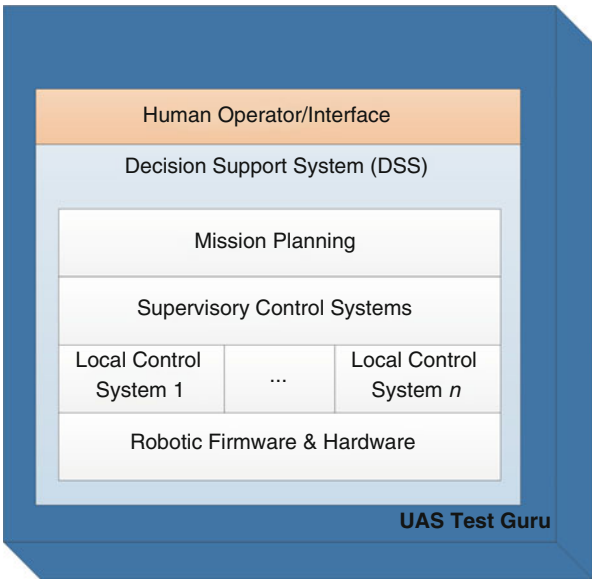

Fig. 7 UAS Test Guru hierarchy (Valerdi 2017)

knowledge gradient algorithms, and Bayesian updating algorithms, which are described below (Valerdi and Blackburn 2009). Furthermore, the DSS can incorporate the human-in-the-loop by considering the collaborative role of humans and robots performing various tasks. The architecture of one such DSS, called the UAS Test Guru (Valerdi 2017), includes a human decision support system interface, high-level motion planner, and supervisory controller which allows for multiple criteria to be evaluated, as shown in Fig. 7.

The objective of the decision support system is to help identify the most critical tests that should be executed to obtain the highest amount of information in the shortest amount of time. This will help transition the test planning activity from a subjective and manual process into a more objective and automated process. We can accomplish this by applying approximate dynamic programming (Bertsekas and Tsitsiklis 1989; Powell 2011) and multi-criteria decision analysis (Keeney and Raiffa 1976; Howard and Matheson 1983) techniques.

Test Optimisation Using Decision Support Systems

Test planning can be extended to include automatic, adaptive, and multi-criteria balancing to enhance the robustness of the method being applied (Valerdi and Enhelder 2016). First, the DSS will accelerate test planning by automating (or partially automating) tasks that are currently human-intensive and error-prone (Valerdi 2017). The DSS will provide automated support for test planning tasks such as test prioritisation, test resource scheduling, and test strategy adaptation, among

others (Valerdi 2017). The main goal is to offer a higher priority to test cases that have better-quality attributes for execution. For example, the DSS will perform test prioritisation by determining the relative value of candidate tests, considering both (1) the predicted importance and utility of the data yielded by each candidate test and (2) the cost of each candidate test, in terms of cost and schedule. Similarly, the DSS will assist test strategy adaptation by proposing ways to improve a set of test plans as more information about a system becomes known. For example, as the undesirable emergent behaviour of an unmanned and autonomous system is uncovered, test plans may need to be modified in order to mitigate the potential risks associated with these undesirable emergent properties.

Second, the DSS can utilise an adaptive planning component algorithm to search for the optimal path of knowledge acquisition (i.e. test sequence) (Valerdi 2017). The algorithm is inspired by traveling salesman and multi-armed bandit problems in operations research (Dayanik et al. 2008; Powell 2011), in which the player (in this case, the tester) has limited knowledge of the system and strives to maximise knowledge acquisition through the optimal sequencing of tests. This approach to testing is fundamentally adaptive in that it aids in constant replanning based on new information obtained from test results. Adaptive algorithms, also known as genetic algorithms, have been applied to a range of system optimisation problems but have not applied to test planning (Hess and Valerdi 2010).

The DSS can optimise test planning by addressing and balancing multiple criteria within a framework based on predetermined preferences. Specifically, the DSS applies formal decision-making techniques such as multi-attribute utility theory (Keeney and Raiffa 1976) to balance various stakeholder preferences. The use of quantitative methods allows the test process to deal more effectively than a human with the inherent complexity of co-robot environments where the many variables and unknowns do not allow "eyeballing" solutions to test planning challenges. In this way, the DSS facilitates the transition from function-based testing of single systems to mission-based testing of co-robot systems. Moreover, by assisting test planners in balancing trade-offs among cost, risk, and schedule when making test planning decisions, the DSS serves a particularly important role in the context of a rapid deployment of systems. The reasoning engine contains the co-robot system specific decision rules that can be adjusted based on user preferences. The outputs of the DSS will be fed to the control supervisor so that replanning tasks can be performed.

The DSS enables test planners to develop and refine a test strategy for co-robot systems that operate in any environment. The test strategies recommended by the DSS address multiple aspects of an overall test plan, including (Valerdi 2017):

- The level of human-robot interaction and the complexity of such tests
- The schedule and order in which to conduct different test events and activities
- The relative importance of various candidate tests, since some tests may need to be omitted due to cost, schedule, or other resource constraints
- The level of effort expected to complete a test plan and its constituent activities

- The resources needed to complete a test plan and the allocation of resources to test events and activities
- The risks associated with a test plan, such as a "domino effect" occurring if a test event fails or a test activity is not completed by a deadline
- The identification of potential undesirable emergent properties
- The options for altering or adapting the test plan as more information (cost, risks, schedule, etc.) is obtained or constraints or goals are changed

Thus, by using the DSS, test planners can create either the template for an initial test plan or refine and improve an existing test plan by incorporating the elements of the test strategy outlined by their stakeholders. For example, test planners might decide to move certain test events earlier in the schedule, eschew some test events due to unacceptable risks, request additional resources needed to gather important data, or reconfigure the test architecture to improve performance (Valerdi 2017).

The test planner's primary interaction with the DSS will be through a dashboard that will provide a user interface through which test planners specify the location of external data artefacts to use as inputs, invoke analysis components, and view test strategy reports containing the outputs of analysis, such as identified risks, recommended test sequences, etc. The dashboard will show what analyses are ready to execute, based on the inputs provided so far, what analyses have already been run, and the effects of any changes to the system configuration can influence test execution. The dashboard can simplify and streamline the human-robot interface of the test bed by enabling more efficient and effective use of resources to support planning-level decisions (Valerdi 2017).

Test Planning Algorithms

Test plans are dynamic entities and allow for the organisation of tests into logical groupings, to minimise redundancy and maximise test efficiency. Through the application of planning algorithms, it is possible to better arrange and schedule relevant system tests by evaluating fault detection as early as possible with minimalised cost and the associated time required for implementation (Oliver et al. 1997). By combining the proposed approaches for local/supervisory control design with a DSS, the validation of such algorithms can be performed much more efficiently and effectively. Smarter, more effective, and more efficient testing of systems can be realised with the help of a test planning tool to facilitate the prioritisation and sequencing of individual tests and composite test sets. These objectives can be accomplished by integrating parametric cost models, knowledge gradient algorithms, and Bayesian updating algorithms (Valerdi and Enhelder 2016).

To date, several adaptive algorithms for manual and automatic/semi-automatic planning have been developed (Hess and Valerdi 2010). However, many fall short in considering the organisational elements and have little respect for the collaborative role of humans and robots performing various tasks. The basis for a test planning algorithm can be described through the following steps (Valerdi and Enhelder 2016):

- Step 1: Prioritise system and/or mission requirements for the system under test. This will be accomplished using a systems-specific implementation of the Stakeholder Win-Win methodology, a multi-criteria preference analysis approach for requirements negotiation.
- Step 2: Define and quantify the cost, c, of running each test. This will be calculated using a *parametric cost model* that considers the complexities of the system under test and the resources (in terms of people, equipment, and facilities) needed to execute each test.

A parametric cost model is a group of cost estimating relationships used together to estimate entire cost proposals or significant portions thereof (ISPA 2008). These models include many interrelated cost estimation relationships, both cost-to-cost and cost-to-non-cost. While cost models have not explicitly been applied to testing in the past, they have been an essential part of product development for a long time. Our own analysis of UAS test events indicates that the most influential cost drivers are *number of systems to be integrated, integration complexity,* and *complexity of tests* as shown in Fig. 8. These technical costs driver scores demonstrate that tests are prioritised according to how complex the system or task is (Deonandan et al. 2010).

- Step 3: Determine $\overline{\theta}_d^o$, the initial estimate of the expected reward for making decision d, where each decision involves selecting a specific test that should be executed. In this case, a reward can be considered to be the generation of new knowledge about the system under test.
- Step 4: Determine $\overline{\sigma}_d^o$, the initial estimate for the standard deviation of $\overline{\theta}_d^o$. The standard deviation is based on the fact that the expected rewards are normally distributed. Higher values of $\overline{\sigma}_d^o$ indicate lower confidence in the decision under consideration.
- Step 5: Execute *knowledge gradient algorithm* to calculate the knowledge gradient (KG) index for feasible decisions. Since the KG jointly optimises three criteria, value, cost, and knowledge acquired, it can be used to develop a prioritisation of the system tests to be performed. At this stage in the process, the first phase of testing is performed and data is collected about the performance of the systems.
- Step 6: Execute *Bayesian updating algorithm* to re-calculate KG index based on new information (e.g. test results, shifting evolving mission requirements, test costs, test facility availability, etc.) and provide an updated test strategy based on the recommended prioritisation of a DSS similar to the UAS Test Guru.

Table 3 illustrates a set of simulated results for an example with five test options (Valerdi 2017). In this case, $\overline{\theta}$ represents the current estimate of the value of deciding to execute each test, while $\overline{\sigma}$ is the current standard deviation of each $\overline{\theta}$. Tests 1, 2, and 3 have the same value for $\overline{\sigma}$, but with increasing values of $\overline{\theta}$. The table illustrates that when variance is the same, the knowledge gradient prefers the decisions that appear to be the most valuable (high $\overline{\theta}$), as indicated by higher KG index score for Test 3. Tests 3 and 4 have the same value for $\overline{\theta}$, but decreasing values for $\overline{\sigma}$,

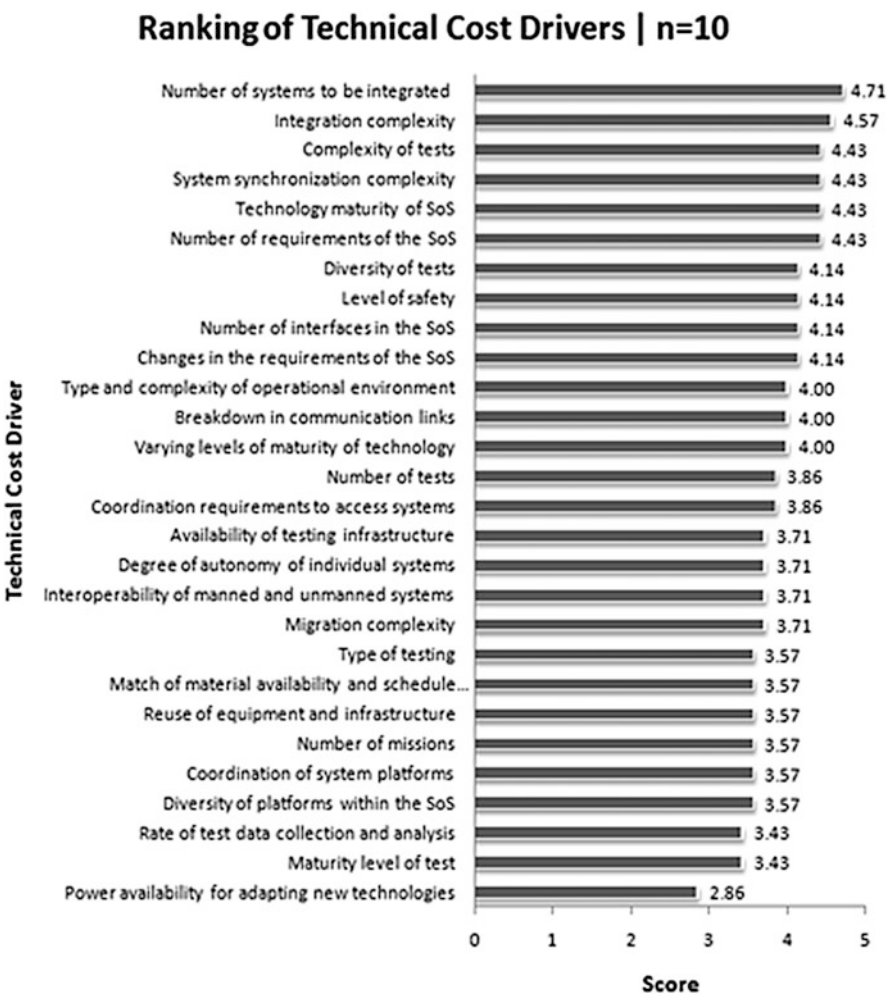

Fig. 8 Relative impact of technical cost drivers for UAS testing. (Deonandan et al. 2010)

Table 3 Knowledge gradient example

Test	$\overline{\theta}$	$\overline{\sigma}$	KG index
1	1.0	1.336	0.789
2	1.5	1.336	1.754
3	2.0	1.336	3.516
4	2.0	1.155	2.467
5	3.0	0.707	0.503

illustrating that the knowledge gradient prefers decisions with the highest variance, as indicated by higher KG index score for Test 3. Finally, Test 5 appears to be the most valuable of all the decisions (high $\overline{\theta}$) but has the lowest variance meaning that

we have the highest confidence in this decision. Therefore, the best decision is to pursue Test 3 since it has the highest KG index score (despite the fact that it is not the most valuable in terms of $\overline{\theta}$).

The systems control algorithms (i.e. control and supervisory) and test planning algorithms when discussed (i.e. cost, knowledge gradient, and Bayesian) together form what is referred to as a human-in-the-loop DSS (reference UAS Test Guru). For the test planner to make the best decision possible, the user interface serves as a DSS, providing the highest amount of information in the shortest amount of time. The DSS as a collection point provides various resources needed to make test planning decisions such as the system(s) under test, cost/risk trade-offs, test progress, and test coupling.

Conclusion

As the demand and development of increasingly complex systems and SoS's continues, testing approaches and DSS will continue to change to improve human experiences and provide ultimately better systems (V&V). In reflecting on the chapter, by better understanding organisational aspects for conducting tests, the phase the test is being performed, the purpose for the test occurring, and the complexity of the system or SoS being tested, a more comprehensive and applicable strategy can be developed. This will help transition test planning activities from a subjective and manual process into a more objective and automated process.

In moving to application and concluding this chapter, the efficacy of the test planning algorithms described, or any of the test planning approaches for that matter, must be evaluated by the following criteria:

- Reduction in test planning time compared to existing manual approaches
- Speed at which test replanning can be done
- Reduction in test schedule through optimised plan provided by the algorithm
- Improved test coverage over time

While, not all of the criteria are equally important or even necessary for all circumstances, the organisational and complexity aspects previously discussed will support more efficient and effective testing. Applications of similar test optimisation methods applied to software testing have shown a 40% reduction in test effort in the financial services industry (Phadke and Phadke 2011) and a 90% reduction in test effort in the telecommunications industry (Cohen et al. 1997) representing millions of dollars of savings. We anticipate equivalent savings in testing engineering systems given similar technical characteristics and complexity.

References

Amland S (2000) Risk-based testing: risk analysis fundamentals and metrics for software testing including a financial application case study. J Syst Softw

Barhorst JF, Paunicka JL, Stuart DA, Hoffman J (2011) Emerging directions in Aerospace Software V&V. Proceedings of Infotech aerospace conference, pp 1507–1512

Bartolomei JE et al (2011) Engineering systems multiple-domain matrix: an organizing framework for modeling large-scale complex systems. Syst Eng 14(3):305–326

Baxter G, Sommerville I (2011) Socio-technical systems: from design methods to systems engineering. Interact Comput 23(1):4–17

Bertsekas DP, Tsitsiklis JN (1989) Parallel and distributed computation: numerical methods. Prentice-Hall, Englewood Cliffs

Blanchard B (2004) System engineering management. John Wiley, Hoboken, p 8

Blanchard BS, Fabrycky WJ, Fabrycky WJ (1990) Systems engineering and analysis, vol 4. Prentice Hall, Englewood Cliffs

Cherns A (1976) The principles of socio-technical design. Hum Relat 29(8):783–792

Clegg CW (2000) Socio-technical principles for system design. Appl Ergon 31(5):463–477. https://doi.org/10.1016/S0003-6870(00)00009-0

Cohen DM, Dalal SR, Fredman ML, Patton GC (1997) The AETG system: an approach to testing based on combinatorial design. IEEE Trans Softw Eng 23(7):437–444

Cohen ML, Rolph JE, Steffey DL (eds) (1998) Statistics, testing, and defense acquisition: new approaches and methodological improvements panel on statistical methods for testing and evaluating defense systems. Committee on National Statistics, National Research Council, National Academies Press, Washington, DC

Cohen J, Plakosh D, Keeter K (2005) Robustness testing of software-intensive systems: explanation and guide. Technical note CMU/SEI-2005-TN-015. Software Engineering Institute, Carnegie Mellon University, Pittsburgh

Coleman DE, Montgomery DC (1993) A systematic approach to planning for a designed industrial experiment. Technometrics 35(1):1–12

Davis FD, Venkatesh V (2004) Toward preprototype user acceptance testing of new information systems: implications for software project management. IEEE Trans Eng Manag 51(1):31–46

Dayanik S, Powell WB, Yamazaki K (2008) Index policies for discounted bandit problems with availability constraints. Adv Appl Probab 40(2):377–400

De Weck OL, Roos D, Magee LC (2011) Engineering systems – meeting human needs in a complex technological world. The MIT Press

Deonandan I, Valerdi R, Lane JA, Macias F (2010, June) Cost and risk considerations for test and evaluation of unmanned and autonomous systems of systems. In: 2010 5th international conference on system of systems engineering, pp 1–6

Department of Defense (DOD) (1983) Life cycle cost in Navy acquisition, MIL-HDBK-259 (NAVY), Washington, DC. 20360

Department of Defense (DOD) (2004) Defense Acquisition Guidebook, Version 1.5, Washington, DC

Elias GM, Jain R (2017) Assessing systems architecture: an exploratory framework. Int J Bus Inf Syst 24(2):127–173

Emery FE, Trist EL (1960) Socio-technical systems. In: Churchman CW, Verhulst M (eds) Management science models and techniques, vol 2. Pergamon, Oxford, UK, pp 83–97

Farr J (2010) Systems life cycle costing: economic analysis, estimation, and management, 1st edn. CRC Press

Farr J (2012) Life cycle cost considerations for complex systems. Systems Engineering – Practice and Theory

Felderer M, Schieferdecker I (2014) A taxonomy of risk-based testing. Int J Softw Tools Technol Transfer 16:559–568

Ferreira S, Valerdi R, Medvidović N, Hess J, Deonandan I, Mikaelian T, Shull G (2010) Unmanned and autonomous systems of systems test and evaluation: challenges and opportunities. IEEE systems conference. 15 pp
Gibson SF (2012) Virtualization of system of systems test and evaluation, naval postgraduate school, NPS-TE-12-003
Gilder G (1993) Metcalf's law and legacy. Forbes ASAP 152(6):158–159
Graves KL (2010) Assessment on how much DoD information technology testing is enough. Naval Postgraduate School
Hess J, Valerdi R (2010) Test and evaluation of a SoS using a prescriptive and adaptive testing framework. 5th IEEE international conference on Systems of Systems Engineering, Loughborough
Hitchins D (2007) Systems engineering: a 21st century systems methodology. Wiley, Hoboken
Hooda I, Chhillar RS (2015) Software test process, testing types and techniques. Int J Comput Appl 111:10–14
Howard RA, Matheson J (1983) The principles and applications of decision analysis (2 volumes). Strategic Decisions Group, Palo Alto
INCOSE (2015) Systems engineering handbook: a guide for system life cycle processes and activities, version 4.0. Wiley, Hoboken
International Society of Parametric Analysts (ISPA) (2008) Parametric estimating handbook, 4th edn. International Society of Parametric Analysts
International-Space-Station-Program-Science-Forum (2015) International Space Station – Benefits for Humanity. Available https://www.nasa.gov/sites/default/files/atoms/files/jsc_benefits_for_humanity_tagged_6-30-15.pdf
Itkonen J, Rautiainen K (2005) Exploratory testing: a multiple case study. 2005 international symposium on empirical software engineering. ISESE 2005
Itkonen J, Mntyl MV, Lassenius C, (2009) How do testers do it? An exploratory study on manual testing practices, Proc. Third Int'l Symp. Empirical Software Eng. and Measurement, pp 494–497
Keeney R, Raiffa H (1976) Decisions with multiple objectives: preferences and value tradeoffs. Wiley, New York
Kerzner H (2017) Project management: a systems approach to planning, scheduling, and controlling. Wiley, Hoboken, New Jersey.
Madani T, Benallegue A (2008) Adaptive control via backstepping technique and neural networks of a quadrotor helicopter. IFAC Proc 41(2):6513–6518
Marais K, Dulac N, Leveson NG (2004) Beyond Normal accidents and high reliability organizations: the need for an alternative approach to safety in complex systems
McQueary CE, Sargeant ST, Nadeau RA, Dunaway DA, Reeves DL, Stephens RC (2009) Using Design of Experiments for operational test and evaluation. Memorandum of Agreement
McShea DW (1996) Complexity and homoplasy, homoplasy: the recurrence of similarity in evolution. Academic, San Diego, pp 207–225
Meinke K (2004) Automated black-box testing of functional correctness using function approximation. Paper presented at the ISSTA 2004 – proceedings of the ACM SIGSOFT international symposium on software testing and analysis, pp 143–153
Montgomery DC (2004) Design and analysis of experiments, 6th edn. Wiley, New York
Mooz H, Forsberg K, Cotterman H (2003) Communicating Project Management
Myers GJ (2011) The art of software testing, 3rd edn. Wiley, New York
NASA (2007) Systems engineering handbook, Revision 1. National Aeronautics and Space Administration (NASA), Washington, DC. NASA/SP-2007-6105
Newman JS (2001) Failure-space: a systems engineering look at 50 space system failures. Acta Astronautica Sci 48(5–12):517–527
Oliver DW, Kelliher TP, Keegan JG (1997) Engineering complex systems with models and objects

Pfeiffer KD, Kanevsky VA, Housel TJ (2011) Mathematical modeling to reduce the cost of complex system testing: characterizing test coverage to assess and improve information return. Naval Postgraduate School, NPS-TE-11-181

Phadke KM, Phadke MS (2011) Utilizing Design of Experiments to reduce IT system testing cost. CrossTalk J Def Softw Eng 24(6):16–21

Philbin S (2008) Managing complex technology projects. Res Technol Manage March–April 51(2): 32–39

Potts M, Sartor P, Bullock S (2020) Assaying the importance of system complexity for the systems engineering community. Syst Eng

Powell (2010) A review of stochastic algorithms with continuous value function approximation and some new approximate policy iteration algorithms for multidimensional continuous applications. J Control Theory Appl 9(3):336–352. https://doi.org/10.1007/s11768-011-0313-y

Powell WB (2011) Approximate dynamic programming: solving the curses of dimensionality. Wiley, New York

Rasmussen J (2003) System design, encyclopedia of information systems. Elsevier

Ryan M, Wheatcraft L (2012) On the use of the terms verification and validation. INCOSE International Symposium

Seglie E (2010) Design of Experiments for test and evaluation. International Test and Evaluation Association Live-Virtual-Constructive Conference

Sheard SA (2013) Systems engineering complexity in context. 23rd annual international symposium of the international council on systems engineering, INCOSE 2013, 2(C), pp 1230–1243

Stephens RC, Herrin RR, Mackenzie D, Knodle DW (2008) Operational Realism via net-centric test & evaluation: from concept development to full-rate production and sustainment. Int Test Eval Assoc (ITEA) J 29(2):147–155

Stockman B, Boyle J, Bacon J (2010) International space station systems engineering case study. Air Force Center for Systems Engineering

Thomas LD (1996) System engineering the international space station. NASA Space Station Program Office

Trist E (1981) The evolution of socio-technical systems: a conceptual framework and an action research program. J Issues Qual Work Life Occas Paper, Issue 2

U.S. National Academy of Engineering Grand Challenges (2021). http://www.engineeringchallenges.org/

Valerdi R (2017) Verification of emergent systems. Unpublished manuscript

Valerdi R, Blackburn C (2009) 6.3.1 the human element of decision making in systems engineers: a focus on optimism. INCOSE International Symposium 19:986–1002

Valerdi R, Enhelder E (2016) Making big data, safe data: a test optimization approach. Defense Technical Information Center, Acquisition Research Program Sponsored Report Series

Wheatcraft L (2012) Thinking ahead to verification and validation. Requirements Experts

Yang Y, He Z, Mao K, Li Q, Nguyen V, Valerdi R (2012) Analyzing and handling local bias for calibrating parametric cost estimation models. Inf Softw Technol

Young RF (2011) Science of test at yuma proving ground. Presentation at Duke University. https://ndiastorage.blob.core.usgovcloudapi.net/ndia/2012/TEST/13684_Johnson.pdf