Received: 13 March 2023 | Revised: 26 May 2023 | Accepted: 19 June 2023

DOI: 10.1002/sys.21709

REGULAR ARTICLE

WILEY

Interpreting integrability

Alejandro Salado

Department of Systems and Industrial Engineering, The University of Arizona, Arizona, USA

Correspondence
Alejandro Salado, Department of Systems and Industrial Engineering, The University of Arizona, Arizona, USA.
Email: alejandrosalado@arizona.edu

Funding information
U.S. Army DEVCOM AvMC

Abstract

Most systems engineers have likely heard of or pursued integrability when developing systems. It is also likely that systems engineers perceive that, in general, integrability is a desired quality to attain. But what is integrability, really? While there is likely a consensus within the systems engineering community about the general idea of integrability, this paper poses that reasoning about integrability requires a deeper, comprehensive interpretation of what integrability is and entails. Recognizing that the meaning of integrability is subjective, this paper provides a deep dive into understanding and interpreting integrability. The findings presented in this paper could be used as a guide for others to find clarity on what they mean by achieving integrability when developing systems.

KEYWORDS
ilities, integrability, non-functional requirements, quality attributes, system integration

1 | INTRODUCTION

The importance of the ilities in system development has been discussed at length in systems and software engineering forums and literature. Here, ilities are understood as outcomes that can yield value to stakeholders by effectively managing the high uncertainty that systems face during their life cycle.[1] Other terms are used to refer to ilities in the literature, such as quality attributes (QA), lifecycle properties, or non-functional requirements in the context of requirements. I use these terms without distinction in this paper.

One such ility that is often critically important in systems engineering is integrability. Being purposefully vague at this point in the paper, integrability refers to certain desired properties of the ability to integrate a system. Given that system integration is central to systems engineering, considering integrability early on in the system development (as with any other ility) is critical because the architecture of a system plays a central role in the extent to which that system will exhibit ilities or lifecycle properties.[2]

Whether ilities are pursued relying on heuristics[3] or analysis,[4] a precise definition of the ility being pursued that avoids conflicting interpretations remains a challenge.[5] This paper explores the concept of *integrability* in depth and shows that existing definitions and ontologies in the literature may be too shallow, failing to capture the richness of the process of integration in real-life applications. The insights obtained in this exploratory study should serve as a guide for others to find clarity and precisely define what they mean when pursuing integrability when developing a system.

This paper is structured as follows. Section 2 presents different interpretations of integrability found in the literature. The theoretical framework used in this paper to guide the study of integrability is presented in Section 3. Section 4 is devoted to dissecting integrability, which is followed by a summary of the different aspects of integrability in the form of a template in Section 5. The main conclusions of this paper are shared in Section 6.

2 | DEFINITIONS AND INTERPRETATIONS OF INTEGRABILITY IN THE LITERATURE

An ility can be defined as the capability of a system to achieve a desired state or outcome.[4] In this sense, integrability might be generally defined as the capability of a system to achieve integration.

There is general agreement in the literature about what system integration is. The ISO[6] defines systems integration as a process consisting of "iteratively [combining] implemented system elements to form complete or partial system configurations in order to build a product or service." Similarly, the Systems Engineering Body of Knowledge (SEBoK) indicates that the ultimate goal of system integration is

to ensure that the individual system elements function properly as a whole and satisfy the design properties or characteristics of the system, explicitly indicating the assembling of elements to form the system is part of system integration.[7] Systems engineering textbooks assume similar interpretations, describing systems integration as "the gradual assembly and testing of the elements that comprise a syste",[8] "the process of assembling the system from its components, which must be assembled from their configuration item",[9] or "integrating the tested components into subsystems and the subsystems into a total operating system [with the objective of] "assembling and integrating the engineered components of the new system into an effectively operating whole".[10] The notion of aggregating components together to form a system is central to the definition of system integration throughout the literature.

Existing definitions of integrability build off this interpretation of system integration, explicitly associating integrability with the ability to make two components work together.[11–13] Some authors further elaborate on the point in time in the system lifecycle in which the two components are made to work together, recognizing that integrability may refer to the moment in which two separately developed components are integrated to form a whole (e.g., literature[11,13,14]) or to the moment in which a component is later integrated into an already existing system (e.g., literature[11,13,15]).

However, theoretical investigations indicate that achieving the desired end state (e.g., achieve integration) is not sufficient to fully describe an ility; the effort it takes to reach that state is also necessary to fully describe the ility.[4] Some authors explicitly refer to this aspect as part of the definition of integrability. For example, Salman[16] defines integrability as the total effort spent on defining intercomponent link and component interfaces, and Jain, Chandrasekaran, Elias and Cloutier[17] refer to "the level of effort required to understand, describe, implement, manage, and document" integration. Yet, the conception that effort to achieve integration also defines integrability is implicit across the literature, as can be inferred from the use of integrability that accompany their definitions. For example, while[14] define integrability as separately developed components working correctly together, they later refer to "the product can be programmatically integrated with other plug-ins with minimum development effort," and refer also to convenience for the end user, ability of many developers working simultaneously without having to go through all source code of a program before starting to contribute, decrease of time developers need to learn the architecture, and ease of integration and substitution of plug-ins. Similarly, other authors comment on integrating quickly and reliably,[18] the amount of change required to integrate,[12] and how much more easily would a new resource be integrated with each other or with an existing system.[13]

3 | METHOD

As shown in the previous section, the definitions of integrability available in the literature essentially state that integrability is the ability to easily make two or more independently developed components work together. However, the literature does not provide clarity in the meanings of *working together* and *easily*.

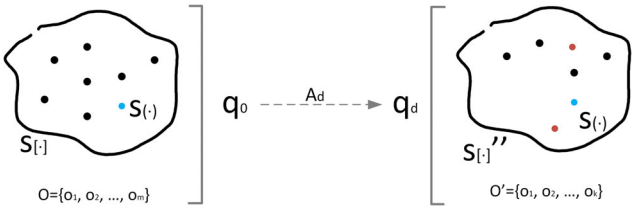

FIGURE 1 Conceptual sketch of a desired situational change [from (Salado, 2022)].

An exploratory investigation of the implications and uses of integrability has been performed, weakly guided by the theoretical framework to conceptualize ilities presented in.[4] By weakly, I mean that the theoretical principles seeded the exploration without constraining it to a specific path of discovery. The exploration used a combination of theoretical derivations, the development of notional examples, and empirical observations from past experiences of the author. The exploration was performed as part of a project aimed at quantifying ilities. Integrability was chosen as the first ility to be studied. Early work showed that existing definitions of integrability were insufficient to fully describe the ility. The goal of the exploration was to show the depth and richness necessary to interpret integrability and was not intended to claim completeness or provide 'the' definition of integrability, since ilities can only be defined by consensus for a specific situation.[4]

To define how ilities are conceived in this paper, I must introduce first a few underlying concepts. I differentiate between the System of Interest (SoI) and the Context System (CS). SoI refers to a system that is put in place with the purpose of satisfying a need or pursuing an opportunity.[19] CS refers to a system formed by the SoI and the set of systems with which the SoI directly interacts[19]; such systems are often referred to as external systems to the SoI. I conceive the SoI as an open system, since it transfers information, matter, and/or energy through its boundaries to interact with external systems.[19] I conceive the CS as a closed system, since only the interactions within it are relevant in this work.[19] An *outcome* is defined as the state of a closed system, which is achieved by the interactions that occur within the closed system.[19] A *capability* is defined as the disposition of an open system (e.g., the SoI) to achieve a desired outcome within a desired closed system (e.g., the CS to which the SoI belongs).

An *ility* of an open system (e.g., of an SoI) is defined as a kind of capability. This is because the open system alone cannot guarantee the desired outcome. In essence, an ility will refer to the disposition of an open system to achieve a desired situational change (ref. Figure 1). A *situation* q is defined as a 3-tuple $q = (s_{(\cdot)}, s_{[\cdot]}, O)$, where $s_{(\cdot)}$ is the open system for which the QA is defined, $s_{[\cdot]}$ is a closed system to which $s_{(\cdot)}$ belongs, and O is a set of outcomes in $s_{[\cdot]}$. I denote the starting situation by q_0. Given a desired situation q_d, there exists a set of actions A_d such that $A_d(q_0) = q_d$. That is, there is a set of actions that, when applied to the starting situation, can yield the desired situation. Furthermore, it is possible to define the effort e_d it takes to execute A_d as a direct function such that $e_d = f(A_d)$, where the type of effort is undefined. Different

systems will enable or facilitate different kinds of actions, which, as a result, will result in different extents to which the ility is achieved.

4 | A DEEP DIVE AT INTEGRABILITY

Several aspects of integrability are discussed next. The sequence in which the aspects are discussed is arbitrary. Many of the aspects were discovered/explored simultaneously; they are presented sequentially for readability.

4.1 | Integration of functions or capabilities

The development of a system can be conceived from the perspective of the outcome it is expected to achieve (generally captured in the form of stakeholder needs) or the functions it should perform (generally captured in the form of system requirements).[19] This framework informs a differentiation of ilities, depending on whether one wants the system to achieve a future situation with respect to the functions it executes or with respect to the capabilities it can achieve. Note that a function transforms inputs into outputs and that a capability is defined as the disposition of a system to achieve an outcome; a capability is hence, as previously discussed, a generalization of the concept of ility.

In the case of integrability, when one states that the system must be integrable, do they mean (i) that the functions of the system can be integrated with the functions of another system, or (ii) that the capabilities yielded by both systems can be integrated. This differentiation is important because one or the other have different implications on how they may be achieved. For example, integrating functionality requires only exploring the compatibility of the two systems being integrated. Integrating capabilities also requires exploring possible emerging conflicts between the use of both systems by external systems. Consider two systems. One system is intended to alert someone (capability) by emitting a light of certain color (function). The other system is intended to spy on someone (capability) by recording their conversation (function). Simplifying the analysis, the two functions are in principle integrable (light will not affect the performance of the voice recording and the recorder will not affect the performance of the light pattern). However, we can guess that their capabilities will likely not be integrable, since the capability of alerting someone directly conflicts with that of spying on them.

This implies that a system cannot be simply qualified as integrable or not; rather, one should explicitly refer to either Functional Integrability or Capability Integrability.[1]

4.2 | Integration as an operational end state or also include process and procedural aspects

Because ilities define dispositions to reach certain end state (outcome), some of the literature limit their definitions and/or models to the conditions necessary for the outcomes to occur at that end state.

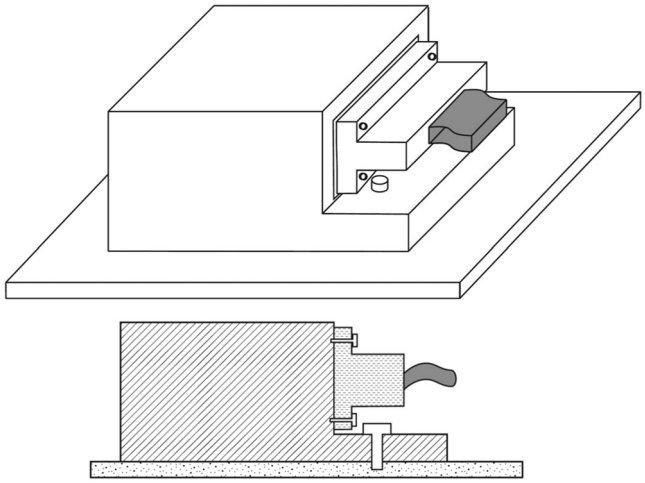

FIGURE 2 Sketch of impossible integration for two compatible elements (bottom: 2D view; top: 3D view).

Consider as an example the definition of integrability provided by the Software Engineering Institute (SEI),[11] which states that integrability is "really about the distance between the elements of each potential dependency—the amount of work needed to resolve differences at each potential dependency." This is then further refined as the two components having semantic equivalence, temporal equivalence, and other characteristics.

This interpretation of integrability focuses on the level of adaptation or change necessary to make two incompatible elements compatible, from the perspective of working together. Therefore, any two elements that are already compatible are equally integrable. This understanding is sensible but potentially insufficient in several cases. Consider for example the connection of two systems depicted in Figure 2. The sketch shows a system formed by a component mounted on top of a plate (System 1) and a second system formed by a cable that connects to System 1 and another component (System 2). Let us assume that the two components communicate with each other using a compatible physical interface (same number of pins, with perfect matching of signals to pins, etc.), using a compatible communication protocol, are syntactically and semantically equivalent, their communication is compatible with the bandwidth of the communication channel, etc. Mechanically, there is space for all pieces of System 2 to integrate with System 1. In other words, once connected, both components work like a charm; zero distance if we look at SEI's definition of integrability.

However, a closer look at the sketch reveals that, while both systems can work together, reaching such state is not possible. Connecting System 2 to System 1 is infeasible because the bolt that fixes the connector (B1) cannot be inserted due to a conflict with the bolt that ties the two components of System 1 (B2). Similarly, removing B2 and placing it back after inserting B1 is also infeasible. The two systems cannot be put together to achieve the "compatible" state.

This example supports the idea that integrability may need to be defined considering characteristics beyond those related to the operational aspects of the desired end state.

4.3 | Intent and usage scenario

The literature primarily treats ilities as inherent attributes of an architecture or a system. For example, authors often try to answer questions such as, *is this architecture integrable?* or, *is it flexible?* without referring to any other contextual condition (there are some exceptions[5]). However, since ilities are defined in the context of a closed system (ref. Section 3) context of usage plays a critical role in the extent with which an ility is exhibited by a system or its architecture. A discussion around integrability follows.

First, let us look at the **specific intent** of an integration scenario, regardless of whether one intends to integrate a capability of a function, since one or more components will need to be integrated in either case. In this case, integrability may refer to:

1. The ability of a component to be integrated into one or more existing systems. Here, integrability is assessed on the component being integrated.
2. The ability of a system to integrate one or more additional components. Here, integrability is assessed on the system.
3. The ability of two or more components to be integrated to form a system. Here, integrability is assessed on the joining of the two components.

Distinguishing between the integration intent is important to assess how architectural choices (or mechanisms[2]) influence integrability. For example, consider topology as an architectural mechanism. For the first case, topology is not an aspect of the component. Since it is not a choice that the architect can make, topology is not an architectural driver in the integrability of the component. This is different in the other two cases, where the topology is an architectural choice in both cases.

Note how this implies that a system architecting effort may actually have to consider the three types of integrability identified above; that is, the three intents. Imagine, for example, that we want to develop a system that (i) is easy to put together (i.e., case 3 above), (ii) can accommodate future components to incorporate new technologies or functionalities (i.e., case 2 above), and (iii) is easy to be integrated in existing and new platforms (i.e., case 1 above). Different mechanisms may exert different influence on the three kinds of integration intents and, therefore, using a single ility to portray integrability may end up being insufficient to inform an adequate or effective architecting effort.

Now, let us look at **different scenarios of integration** that share the same integration intent. In this case, while the integration intent is the same, there are different scenarios in which integration occurs and each scenario has different expectations. As a result, the assessment of the integrability of the system yields different results. This is critical for modeling and assessing ilities because it indicates that different *variations* of integrability interpretation may be necessary as part of an architecting effort.

This finding derives from an actual engineering project, which is kept sanitized for confidentiality. Simplistically, the system consisted of several boxes assembled together and the main users of the system were

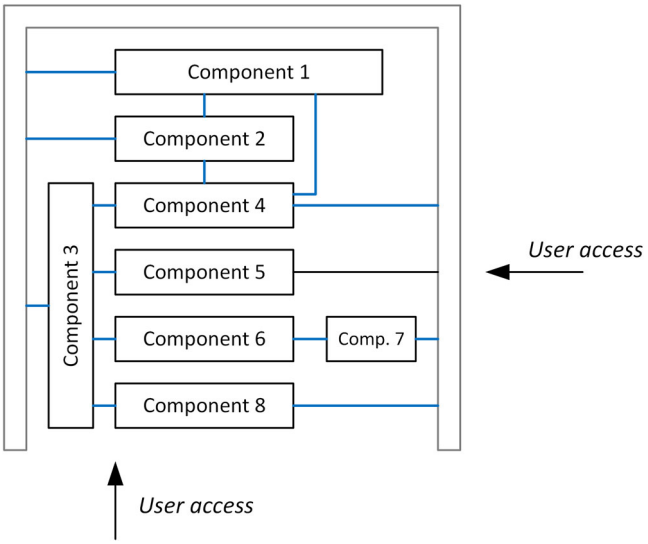

FIGURE 3 Sketch of integration example (with footnote 3).

astronauts. The integration intent was Case 3 in the previous example. That is, integrability was assessed based on the ease with which the different components that formed the system could be put together to form the system. Such integration effort was designed to occur in space and carried out by the astronauts.

Developmental testing indicated that test engineers assessed the system to be poorly integrable. Because the components were connected and mounted onto each other following a sort of electrical and mechanical daisy chain (see sketch in Figure 3), the debugging of virtually any error detected during integration testing required pulling apart most of the components that formed the system. At the same time, almost every integration test required all components to be integrated. Furthermore, integration and de-integration could only be done with a specific tool and following one specific sequence of actions. As a result, time to integrate was high and there was no flexibility of action to support integration, among others, which again, were considered by test engineers as symptoms of poor integrability.

However, integration by the users (astronauts) told a different story. The whole integration only required using one tool that could be handled with a single hand. Mounting units on top of each other avoided by design the possibility that the user would misconnect the elements. Furthermore, a single person could do the complete integration, without requiring external tools to control for alignment and avoiding the injection of extraneous forces and similar aspects when integrating the different components. Finally, cognitive effort was extremely low, and the process was robust because of a rigid sequence of integration events that did not allow for changes by design. From a user perspective, the system was highly integrable.

This case shows that the same system with the same integration intent and the same integration sequence is perceived as both poorly and highly integrable by different stakeholders, depending on the context in which they interacted with the system.

Overall, this finding indicates a richer taxonomy of integrability, which requires capturing intent and usage context to completely define the ility.

4.4 | Symmetry

One of the cases in the previous section hints at de-integrating (or disassembling) components as something a type of users considered to be related to integrability. This bears the question of whether ilities are symmetric or not. In other words, given that an ility is defined as a disposition to reach a situation or state, does the ility also imply the same level of disposition to revert to the original situation or state? For example, if a system is said to be integrable, does that imply that the system is both easy to put together and easy to take apart?

Engineering practice and literature indicate that ilities in general, and integrability in particular, are implicitly assumed to be symmetric; it suffices to do a quick literature search on de-ilities such as de-integrability, de-portability, de-configurability, de-flexibility, or similar, and see the scarcity of results. However, there are plentiful counterexamples in engineering practice that demonstrate that at least integrability is not necessarily symmetric. For example, while screwing and unscrewing a screw are similarly easy, a finish nail is very easy to insert and quite difficult to de-insert. Similarly, integrating a USB drive to a computer requires just one action (plug the drive) but de-integrating it requires three actions (right click on USB devices, choose to unplug, unplug).

Explicitly addressing symmetry when defining integrability is important to align architectural choices with integration (and de-integration) needs. While some systems may need to be easy to integrate and de-integrate, there may be situations that require a system to be easy to integrate but difficult to take apart or vice versa. Similarly, while there may be architectural mechanisms that support integrability and de-integrability at once, there may architectural mechanisms that support one integrability or de-integrability at the expense of the other one.

Since symmetry is not guaranteed, the definition of integrability must also incorporate this distinction.

4.5 | Directionality and integrated behavior

One approach to define integrability consists of identifying characteristics that result in integrability. For example, the SEI states that integrability results from achieving syntactic equivalence, data semantic equivalence, behavioral semantic equivalence, temporal equivalence, and resource equivalence.[11] However, the universality of a set of characteristics that lead to integrability is questionable. In fact, as discussed in this section, characteristics may differ depending on the end-goal of the desired integrability, where end-goal is different from intent and scenarios discussed earlier.

Since a system (or a component) is developed with the objective that it provides a function (or enables a capability), it makes sense to claim that the purpose of integration is to yield an integrated functionality or capability.[3] This applies regardless of integration intent, that is, of whether one wants to integrate two components or a component to an existing system, for example. Therefore, integrability becomes, in essence, the ability to integrate functions (and maybe we could add *that are performed by distinct components*) or capabilities. Integrating functions (or capabilities) can be done with different end-goals in mind. For example:

1. When the two functions (or capabilities) are different and not intended to be composed as a *more sophisticated* function (or capability), to:
2. Save resources (e.g., memory, energy, volume, etc.).
3. Ease testing, maintenance, etc.
4. When the two functions (or capabilities) are the same, to:
5. Increase capacity (e.g., energy, communication bandwidth, computational power, etc.).
6. Increase robustness, availability, and so forth (e.g., using functional redundancy).
7. When the two functions (or capabilities) are different and intended to be composed, to:
8. Provide a more sophisticated functionality (or capability).

To explore characteristics relevant to integrability, one needs to find out what conditions need to be met for integration to be successful. To ease the discussion, let us focus on functional integration. To the most fundamental level, integrating functionality implies that both functions correctly operate together. Therefore, a fundamental condition of functional integrability is that the execution of one function does not jeopardize the other one; in other words, the operation of one function cannot disturb the operation of the other function. Regardless of the type of functional integration, with one possible exception, this implies the following:

1. The environmental characteristics that one of the components generates need to result in an environment that is compatible with the boundaries of operation of the second component. I use the term *result* because compatibility depends on the emergent environment, not the contribution of the individual component. This condition implies that all environmental properties need to be considered. Examples include electromagnetic environment, electromagnetic shock, thermal flows, surface roughness, mechanical vibrations, optical and audio emissions, humidity, pressure, particulate contamination, center of gravity, moments of inertia, etc.
2. The resources consumed by one component to execute its function must leave sufficient resources for the second component to execute its own function. This condition implies that all types of resources need to be considered. Examples include all types of energy and power, memory, buffers, computational capacity, input and output ports, volume, mass, bandwidths, electromagnetic spectrum, timing and commanding, etc.
3. The operational needs of one component must not alter the operational needs of the other component. Examples include security

TABLE 1 Conditions for integrating identical functions.

Element	Description	Necessary condition for integration
Inputs		
Common	The same input drives both functions.	The "loading" of one component must not alter the input interpretation of the other one.
Splitting	A single input is divided and each part feeds one of the functions.	Common loading of the input must not yield input loss nor overlapping.
Independent	Inputs are allocated by filling incapacity.	No additional condition seems needed in this case.
Outputs		
Aggregated	The outputs of both functions are added up.	Different types of aggregation may be sought (e.g., coherent, incoherent). They will require different conditions, also depending on the way the output is generated and conveyed. More on this later.
Independent	The outputs of each function remain independent.	No additional condition seems needed in this case.
Functions		
Simultaneous operation	Both functions are executed at the same time.	I do not think that additional conditions are necessary.
De-phased, independent operation	Just one of the functions is executed at a given time, and the past execution of one has no effect on the current execution of the other.	Synchronism is necessary.
De-phased, dependent operation	Just one of the functions is executed at a given time, but the past execution of one may affect the current execution of the other	Synchronism is necessary, as well as a history tracking.

and vulnerabilities, safety, sustainment, sequence of operations, behavioral synchronicity, etc.

The exception occurs when the purpose of integration is destructive instead of synergistic. Whereas in synergistic integrability there is a desire for the integrated systems to work correctly, in destructive integrability just one system needs to operate correctly. Destructive integrability may be desired, for example, with systems that are intended to attack other systems. The system must be integrable in a variety of systems so that it can operate in the victim system, but its objective is to disturb the victim. Therefore, compatibility, as identified earlier, is only required in one direction. In this sense, we can already see another distinction for integrability, which can be conceived as one-directional or bi-directional.

Continuing with synergistic integrability, while the characteristics identified earlier are ubiquitous, the purpose of the functional integration leads to additional yet possibly different conditions to achieve integrability.

Type 1. Different functions, not composed.

Because, in this case, the functions are not composed in any way, but remain independent and are different, no other condition is necessary to guarantee integrability.

Type 2. Identical functions.

Let us consider the different conditions by which two identical functions may be desired to be integrated, and therefore, the boundaries of the functions and the functions themselves (ref. Table 1). This structure can be used to identify conditions that are necessary for integration to succeed.

These conditions can be further refined. For example, the conditions to guarantee the required output aggregation may depend on the type of output that the functions provide. Any output of a function can be modeled as a combination of energy, material, and information.[10] Using a generic communication model to structure the different characteristics that the output may exhibit, conditions related to the channel through which the output is transferred apply to all kinds of outputs and those related to the message also apply to information outputs. For example, in this case, coherent aggregation may need channel equivalence (i.e., compatibility of the physical aspects of the interface such that two systems can be connected, for example, port equivalence, electrical properties equivalence, communication protocol equivalence, etc.), synchronicity, etc.

Type 3. Different functions, composed.

Composition of functions implies exchanging items between functions. These items can be modeled as a combination of material, energy, and information, as previously discussed. Fundamentally, one function receives and uses a combination of material, energy, and information provided by the other function. Furthermore, the two functions do not only need to exchange those items, but also do so in a coordinated manner. One can look at what is needed for successfully exchanging each type. In addition to those identified in the previous case, the connection needs material equivalence (i.e., the receiving function can process the received material and the sent material is sufficient for the receiving function's needs), energy equivalence (i.e., the receiving function can process the received energy and the sent energy is sufficient for the receiving function's needs), and semantic equivalence.

It should be noted again, however, that these conditions assume a specific desired functional composition. There are desired functional integrations that would in fact call for the opposite conditions, such as a system with the purpose to jeopardize another one (in which case, its attack may be based on leveraging inequivalent material, energy, and/or information, or saturating or breaking channel equivalence) or a system that will integrate with another one to relay classified information, but there is the intention that the connected system is not able to understand the content of the message (in which case, success of integration requires semantic inequivalence).

4.6 | Success criteria

Because integrability has been defined as a disposition, a first thought is to characterize it as a likelihood. For example, one might characterize integrability as the likelihood to achieve the desired integration. A question that arises is whether the likelihood of exhibition (i.e., of achieving integration) is a sufficient metric or not for the purpose of architecting a system.

Likelihood of exhibition means that the system either exhibits integrability or it does not. For example, this would imply that a system either is integrable, or it is not. In other words, it considers integrability to be a binary quality. An alternative is to consider an *extent* to which integrability is exhibited. This means that, when a system is not integrable, we distinguish between those that are *far* from being integrable and those that are *almost* integrable. Using the first interpretation adds an understanding to integrability such that the more systems a system can work with directly, the more integrable it is. The second interpretation adds an understanding such that the lower the average work necessary to make the system work with as many systems as possible, the more integrable the system is.

Because these two interpretations can effectively lead to different architectural choices, the success criteria of integrability must be part of its definition. In essence, the definition of integrability must answer the question, which system is more integrable, one that directly works with 80% of the systems (no change required) but requires investing $1 M to work with the other 20%, or one that directly works with 20% of the systems (no change) but requires $200K to work with the rest.

Now, scoping success as a factor of the work needed to achieve integration may be limiting. This has been referred to in the literature as easiness to achieve integration, which was hinted at in Section 3, but easiness may refer to various aspects. For example, time or money necessary to achieve integration, dependency on external actors to carry on the integration (such as tools or personnel), process dependency (i.e., how many different paths one could exercise to achieve the desired end state), waste that needs to be disposed of or additional material that is necessary, absence of safety problems, resulting likelihood of rework, extent of repetition of tasks, or cognitive effort necessary to carry on integration. These aspects relate to different needs that may be sought by pursuing integrability and, therefore, they must be part of the definition of integrability.

TABLE 2 A template to define integrability.

Element	Values
Object	Functions capabilities
Desired behavior	Different functions (or capabilities), not composed Identical functions (or capabilities) Different functions (or capabilities), composed
Intent	Parts into a whole Of a component to a system Of a system to components
Usage scenario	Actor and context
Symmetry	Symmetrical Asymmetrical
Activity	End state Process/procedural
Directionality	One-directional (destructive) Bi-directional (synergistic)
Success criteria	Probabilistic target & Extent Desired aspect

5 | A TEMPLATE OF INTEGRABILITY

The results of the exploratory study presented in the previous section suggest that defining integrability requires addressing at least the different elements listed in Table 2. In fact, different integrabilities may be defined for a given system or in a given system development context. This is a fundamental novelty with respect to the literature, where integrability is conceived as a single attribute of a system. Instead, the results of this paper indicate that different kinds of integrability may be defined for a given system, noting that the values of some of the elements may be mutually exclusive for a given object but can coexist for different objects, and different objects may be treated differently within a system development context.

Two examples of the application of the template follow. It should be noted that the values and descriptions in real-life applications will likely be more exhaustive, but they are consider sufficient to show how integrability ought to be defined.

5.1 | Example 1

Consider an optical instrument formed by a telescope, a spectrometer, and a camera mechanically sustained by a structure with the objective to image a target. The desired integrability is defined in Table 3.

5.2 | Example 2

Consider a computer system built of a keyboard, a processing system, and a display. The computer system is intended to provide computational resources and expected to integrate with future devices (e.g., a camera, a sound recorder…). In this case, two kinds of integrability need to be defined, one addressing the integration of the system and one addressing the integration of future devices. They are defined in Tables 4 and 5, respectively.

TABLE 3 Definition of integrability for Example 1.

Element	Value	Description
Object	Functional integration	*Optical instrument*: Light to digital data w/ specific performance (e.g., resolution, MTF)
Desired behavior	Different functions, composed	*Telescope*: Collect light (Light rays to concentrated light) w/ specific performance *Spectrometer*: Separate incoming light into spectral bands w/specific performance *Camera*: Convert light into digital data w/ specific performance *Structure*: maintain components in certain position w/specific performance $F_{OPTICAL\ INSTRUMENT} = f(F_{TELESCOPE}, F_{SPECTROMETER}, F_{CAMERA}, F_{STRUCTURE})$
Intent	Parts into a whole	Telescope, spectrometer, camera, and structure integrated to form the optical instrument.
Usage scenario	Operator in clean room	Operator/s integrate the components in the clean room using a specific set of tools and equipment.
Symmetry	Symmetrical	Symmetry is sought because there is a risk of malfunction at instrument level that requires de-integration.
Activity	End state & Process/procedural	Integration is achieved once the functionality of the optical instrument is confirmed but the integration procedure contributes to both the instrument functionality and the success criteria of integration.
Directionality	Bi-directional (synergistic)	The functions of telescope, spectrometer, camera, and structure must correctly operate for the optical instrument to correctly operate.
Success criteria	Probabilistic target & Extent Desired aspect	Minimize time, minimize risk of rework, robust against unavailability of any component, no more than 2 operators necessary…

TABLE 4 Definition of integrability 1 for Example 2.

Element	Value	Description
Object	Functional integration	*Computer system*: Collect inputs and provide results.
Desired behavior	Different functions, composed	*Keyboard*: Collect inputs and convert them to digital commands w/ specific performance *Processing system*: Perform required operations on inputs and provide results as digital data w/specific performance *Display*: Provide results in human readable format w/ specific performance $F_{COMPUTER\ SYSTEM} = f(F_{KEYBOARD}, F_{PROCESSING\ SYSTEM}, F_{DISPLAY})$
Intent	Parts into a whole	Keyboard, processing system, and display integrated to form the computer system.
Usage scenario	Operator in clean room	Automated integration in a production facility.
Symmetry	Asymmetrical (only forming whole)	The computer's hardware is not designed to be changed. Product is mass produced with no need for rework/repair. Integrability only necessary to put the computer together.
Activity	End state & Process/procedural	Integration is achieved once the functionality of the computer system is confirmed but the integration procedure contributes to the success criteria of integration.
Directionality	Bi-directional (synergistic)	The functions of keyboard, processing system, and display must correctly operate for the computer system to correctly operate.
Success criteria	Probabilistic target & Extent Desired aspect	Minimize time and cost

6 | CONCLUSIONS

This study has shown that there is not one kind of integrability. Rather, integrability may be interpreted in many different ways. Furthermore, different integrabilities may be even defined for a given system or in a given system development context. This is a fundamental novelty that contrasts with existing literature, which assumes a common understanding of integrability and treats it as a single attribute of a system.

The variety with which integrability may be interpreted makes it necessary to attain a richer understanding of the integration conditions that are sought if one wants to adequately define integrability. Without aiming at being complete or exhaustive, this study has provided several elements that must be addressed when defining integrability to achieve precision. These include the object of integration (whether one wants to integrate functions or capabilities), the sought behavior of the integration (e.g., redundancy of functions or composition of functions into a more sophisticated one), its intent, the scenario in which integration is expected to occur, the need for symmetry and/or specific directionality, and the way in which integration success is determined. Different values of each one of those elements will yield a different definition and interpretation of integrability.

TABLE 5 Definition of integrability 2 for Example 2.

Element	Value	Description
Object	Functional integration	Computer system with external device/s (called here Integrated system).
Desired behavior	Different functions, composed	*Computer system*: Collect inputs and provide results. *External devices*: Several possibilities (e.g., collect visual information, collect sound information, provide results using sound, store information…). $F_{INTEGRATED\ SYSTEM} = f(F_{COMPUTER\ SYSTEM}, F_{EXTERNAL\ DEVICE})$
Intent	Of a system to components	Future external devices into existing system.
Usage scenario	User in operational context	User integrates the external device into the computer system.
Symmetry	Symmetrical	There is an expectation that the user will integrate and de-integrate multiple external devices.
Activity	End state & Process/procedural	Integration is achieved once the functionality of the integrated system is confirmed but the integration procedure contributes to the success criteria of integration.
Directionality	Bi-directional (synergistic)	The functions of computer system and the external device must correctly operate for the integrated system to correctly operate.
Success criteria	Probabilistic target & Extent Desired aspect	Minimize time, minimize risk of damaging computer system and/or external device, minimize cognitive effort (including learning).

The value of facilitating richer definitions of integrability is likely obvious for both the practitioner and the researcher. For the practitioner, increased precision in the definition of integrability may lead to increased agreement during need elicitation and/or requirement formulation and evaluation of verification and/or validation evidence. For the researcher, increased precision in the definition of integrability may help in narrowing findings in related research. To support this, a summary template has been presented as a guide for others to find clarity on what they mean by achieving integrability when developing systems.

This exploratory study has also contributed to expanding the understanding of the conditions necessary to achieve integration, which serve as proxies for those that lead to integrability. These have shown that, contrary to existing literature, the conditions may differ depending on the specific interpretation of integrability.

Finally, some of the discussion has hinted at the potential of the elements presented in this paper to be generalizable to other ilities. This is suggested as a path of future research.

ORCID

Alejandro Salado https://orcid.org/0000-0001-9378-0795

ENDNOTES

[1] Note that many of the aspects addressed in this paper might transfer or generalize to other ilities. For example, in the case of portability, for a system that can inflate balloons at room temperature (capability) by blowing air (function), it is not the same to integrate the functionality in a room on fire (air blowing is unaffected, assuming compatibility of other functional conditions) than its capability (balloons will unlikely inflate if there is fire… they will melt). The paper is limited though to integrability and the generalization of some of the findings are left for future work.

[2] Architectural mechanism is defined as in 4. Salado A. An experimentation framework for validating architectural properties as proxies for the ilities. *Systems Engineering*. 2022;25(4):342-359. doi:https://doi.org/10.1002/sys.21618. The term mechanism is used in the systems engineering literature with many meanings. Among others, it may refer to the design of incentives in teams (from mechanism design, ref. 20. Vermillion SD, Malak RJ. An investigation on requirement and objective allocation strategies using a principal–agent model. Ibid.2020;23(1):100-117. doi:https://doi.org/10.1002/sys.21511), ways in which change occur and unfold (ref. 21. Ross AM, Rhodes DH. *Anatomy of Change Mechanism*. 2011. , 21. Ibid.), to moving parts in a system, to ways in which a failure occurs (ref. 22. Engel A, Teller A, Shachar S, Reich Y. Robust design under cumulative damage due to dynamic failure mechanisms. *Systems Engineering*. n/a(n/a)doi:https://doi.org/10.1002/sys.21588), and to tactics and patterns used in architecture (ref. 11. Kazman R, Bianco P, Ivers J, Klein J. *Integrability*. 2020. CMU/SEI-2020-TR-001.). We take the latter meaning of the term from the work in system architecture performed by the Software Engineering Institute (SEI).

[3] This is the case even if we think about the integrability of a component that performs a function (or capability) to jeopardize, destruct, or compromise the function (or capability) with which it will integrate. This situation can still be considered an integrated functionality (or capability). It is a type of functional (or capability) composition, which will be explained later.

ACKNOWLEDGEMENTS

The author thanks R. Gerlach and B. Zane for producing Figure 2. This paper reports work performed as an independent consultant to the U.S. Army through Skayl, LLC and not as an employee of the University of Arizona. Effort sponsored in whole or in part by the U.S. Army Aviation and Missile Research, Development and Engineering Center, under Recipient Agreement No. W911W6-17-3-0002. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S. Army Aviation and Missile Research, Development and Engineering Center.

DATA AVAILABILITY STATEMENT

None.

REFERENCES

1. Mcmanus H, Hastings D. A framework for understanding uncertainty and its mitigation and exploitation in complex systems. *IEEE Eng Manage Rev*. 2006;34(3):81. doi:10.1109/EMR.2006.261384

2. Osorio CA, Dori D, Sussman J. COIM: an object-process based method for analyzing architectures of complex, interconnected, large-scale socio-technical systems. *Syst Eng*. 2011;14(4):364-382. doi:10.1002/sys.20185
3. Maier MW, Rechtin E. *The Art of System Architecting*. CRC Press; 2009.
4. Salado A. An experimentation framework for validating architectural properties as proxies for the ilities. *Syst Eng*. 2022;25(4):342-359. doi:10.1002/sys.21618
5. Ross AM, Rhodes DH. Towards a prescriptive semantic basis for change-type ilities. *Procedia Comput Sci*. 2015;44:443-453. doi:10.1016/j.procs.2015.03.040
6. ISO. ISO/IEC/IEEE International Standard - Systems and software engineering – System life cycle processes. In: ISO/IEC/IEEE 15288 First edition 2015-05-15. 2015:1-118. doi:10.1109/IEEESTD.2015.7106435
7. Snoderly J, Faisandier A, Jackson S. System integration. SEBoK; 2021.
8. Sols A. *Systems Engineering. Theory and Practice*. Universidad Pontificia de Comillas; 2014.
9. Buede DM, Miller WD. *The Engineering Design of Systems: Models and Methods*. 3rd ed. John Wiley & Sons, Inc.; 2016.
10. Kossiakoff A, Sweet WN, Seymour SJ, Biemer SM. *Systems Engineering Principles and Practice*. 2nd ed. John Wiley & Sons, Inc.; 2011.
11. Kazman R, Bianco P, Ivers J, Klein J. Integrability. CMU/SEI-2020-TR-001. 2020.
12. Ribeiro L, Hochwallner M. On the design complexity of cyberphysical production systems. *Complex*. 2018;2018:1-13. doi:10.1155/2018/4632195
13. Farid AM. Measures of reconfigurability and its key characteristics in intelligent manufacturing systems. *J Intell Manuf*. 2017;28(2):353-369. doi:10.1007/s10845-014-0983-7
14. Henttonen K, Matinlassi M, Niemela E, Kanstren T. Integrability and extensibility evaluation from software architectural models – a case study. *Jopen source softw*. 2007;1:1-20.
15. Schulz AP, Fricke E, Igenbergs E. 8.4.2 Enabling changes in systems throughout the entire life-cycle – key to success? *INCOSE International Symposium*. 2000;10(1):565-573. doi:10.1002/j.2334-5837.2000.tb00426.x
16. Salman N. Complexity metrics as predictors of maintainability and integrability of software components. *Çankaya univ j sci eng*. 2006;1(5):39-50.
17. Jain R, Chandrasekaran A, Elias G, Cloutier R. Exploring the impacts of systems architecture and systems requirements on systems integration complexity. *IEEE Syst J*. 2008;2(2):209-223.
18. Mehrabi MG, Ulsoy AG, Koren Y. Reconfigurable manufacturing systems and their enabling technologies. *Int J Manuf Technol Manage*. 2000;1(1):114-131. doi:10.1504/ijmtm.2000.001330
19. Salado A. A systems-theoretic articulation of stakeholder needs and system requirements. *Syst Eng*. 2021;24:83-99. doi:10.1002/sys.21568
20. Vermillion SD, Malak RJ. An investigation on requirement and objective allocation strategies using a principal–agent model. *Syst Eng*. 2020;23(1):100-117. doi:10.1002/sys.21511
21. Ross AM, Rhodes DH. Anatomy of change mechanism. 2011.
22. Engel A, Teller A, Shachar S, Reich Y. Robust design under cumulative damage due to dynamic failure mechanisms. *Syst Eng*. 2021;24:322-338. doi:10.1002/sys.21588

How to cite this article: Salado A. Interpreting integrability. *Systems Engineering*. 2024;27:99–108. https://doi.org/10.1002/sys.21709

AUTHOR BIOGRAPHY

Dr. Alejandro Salado has over 15 years of experience as a systems engineer, consultant, researcher, and instructor. He is currently an associate professor of systems engineering with the Department of Systems and Industrial Engineering at the University of Arizona. In addition, he provides part-time consulting in areas related to enterprise transformation, cultural change of technical teams, systems engineering, and engineering strategy. Alejandro conducts research in problem formulation, design of verification and validation strategies, model-based systems engineering, and engineering education. Before joining academia, he held positions as systems engineer, chief architect, and chief systems engineer in manned and unmanned space systems of up to $1B in development cost. He has published over 100 technical papers, and his research has received federal funding from the National Science Foundation (NSF), the Naval Surface Warfare Command (NSWC), the Naval Air System Command (NAVAIR), and the Office of Naval Research (ONR), among others. He is a recipient of the NSF CAREER Award, the International Fulbright Science and Technology Award, the Omega Alpha Association's Exemplary Dissertation Award, and several best paper awards. Dr. Salado holds a BS/MS in electrical and computer engineering from the Polytechnic University of Valencia, a MS in project management and a MS in electronics engineering from the Polytechnic University of Catalonia, the SpaceTech MEng in space systems engineering from the Technical University of Delft, and a PhD in systems engineering from the Stevens Institute of Technology. Alejandro is a member of INCOSE and a senior member of IEEE and AIAA.