Doc No. TBD

System Architecting

Version 0.92 (initial release for review)

Prepared by: Robert Rasmussen, John Day, Steven Jenkins

May 26, 2020

Jet Propulsion Laboratory
California Institute of Technology

Change Log

Version	Release Date	Description
Initial draft	2019 Oct 24	Initial draft with a few items to be completed, as noted in comments
Initial release for review	2020 Apr 3	Primary content, but with deferrals, as indicated in Word-version comments

■

This document was prepared at Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration (80NM0018D0004)

Table of Contents

Appendices 53

■

1 Introduction

A methodology is a system of processes, techniques, notations, supporting tools, and other means used in the performance and assessment of some activity. The purpose of this document is to introduce a methodology that can be used effectively to perform system architecting on JPL flight projects.

This proposed methodology has been developed under the auspices of JPL's Integrated Model-Centric Engineering Initiative (IMCE) with support from the JPL Project Support Office. [1] It differs from current practice in a few essential respects:

- It introduces and employs a formal vocabulary for system architecting knowledge that is aligned with recognized best practices in architecting.
- It employs mathematical formalism to make both pertinent facts and architectural decisions explicit and rigorous.
- It lends itself to supporting human creativity with automated analysis.
- It facilitates our ability to carry information and expertise across technical domains and from project to project.

Such ideas are addressed in this architecting methodology through three basic components:

- A **framework** that defines a normative structure within which an architecture description can be captured and systematically manipulated, analyzed, and used;
- A set of **viewpoints** that cover most areas of interest routinely handled by JPL projects during development, generally organized by recurring concerns, with guidance on view content and reusable abstractions for their instantiation in views; and
- A **process** for applying general patterns of architectural elaboration over time and at successive development stages, with identified criteria for successful completion.

Following is a description of these three methodology components, followed by a description of the sorts of **tools**, **training**, and other support capabilities needed for their application.

■

2 Framework

During every design effort, many competing factors are in play. The aim of architecture is to find and establish an achievable balance among them that ensures a satisfactory outcome for all concerned. However, interdependencies among the many aspects of a design can be extraordinarily intricate, especially when objectives are ambitious, but constraints are strict. Therefore, complete and precise communication is vital, both among developers and with those who define success. Establishing accepted value to stakeholders, unambiguously delineating responsibilities within a principled structure, making broad allowances for evolving implementation, and carefully articulating interfaces can assure the confident orderly progression of subsequent development and operation. This is the essence of good architecture.

Documents have long served as the primary vehicle for communicating architecture, in its most formal sense. In documents, we say what we intend to do, and we explain why we think this is right. Moreover, we declare that no documents, other than a sanctioned set, have this authoritative status, attempting in this way to minimize what needs to be correct and to resolve concerns over conflicting information. That is, we try to be of one mind—our documents asserting a shared clarity of purpose and understanding—the reward of good communication.

Given their importance, attaining the completeness and stability of sanctioned documents has been a pivotal aspect of the design process. This is not easy. An encyclopedic rendering of complex design demands an organizing structure that fosters lucid separation of concerns while rigorously heeding the ties that bind them. These considerations mirror the aspirations of elegant architecture, so in this endeavor, documents can serve to portray and preserve valuable architectural ideas.

Nonetheless, in this role, documents have so far been mostly passive artifacts, relying by their nature on external processes for interpretation and consensus. Naturally therefore, documents are often viewed more as the endpoint or record of some process than as the medium of its accomplishment. For the latter, we rely instead on other methods, whether in CAD drawings, spreadsheets, simulations, or otherwise, that provide the formality of actionable expression generally missing from conventional documentation. These in turn are too often mediated through ad hoc, informal, and transitory processes, and they tend to direct attention away from architecturally guided implementation toward meandering point design instead.

The purpose of a model-based framework for architecture is to narrow this communications gap by introducing more formal structure into the rendering of architecture than documents alone can accomplish. Information expressed in this manner would in principle be directly amenable to automated analysis and beneficial to interoperability among other design activities, thereby enabling iteration within the architecture model as a productive work environment.

In its full expression, this approach would necessitate the adoption of an extensible information model, such that the narratives we write to explain our intentions would be merely supplemental to underlying formal content where primary meaning is asserted. However, the **architecture framework** adopted here does not necessitate so drastic a leap. Instead, the narrative form is taken as a peer to formal representations, but in a manner that encourages the formal structure

to reveal itself in that narrative. In this way, formalism acts in the service of narrative clarity, not as a distraction from it—the hallmark of good scientific and engineering communications.

This simplified framework is described below. It employs terminology, ideas, and techniques that are familiar across the architecting community and that appear broadly in other architecture frameworks, view models, and so on [1]. Choices here have been made in favor of broad, easy introduction into JPL projects, but with an ability to accommodate advances in modeling formalism.

2.1 Views

The coarse structure of this architecture framework is most evident in the organization of its architectural **views**. A view is analogous to a document, in that it addresses, in a combination of narrative and logical forms, a particular topic or discipline of interest.

Views are related to one another through a **view hierarchy** that establishes, in a manner analogous to a document tree, an ordering of precedence. The assertions made within any view necessarily constrain those made by responding views, permitting top to bottom **traceability** in a fashion analogous to requirement flow-down. The views embodying this flow constitute **rationale** for successive elaborations (e.g., functional decompositions), terminating at the point where details have been safely encapsulated within distinct, specified products for which handoff to independent development is possible.

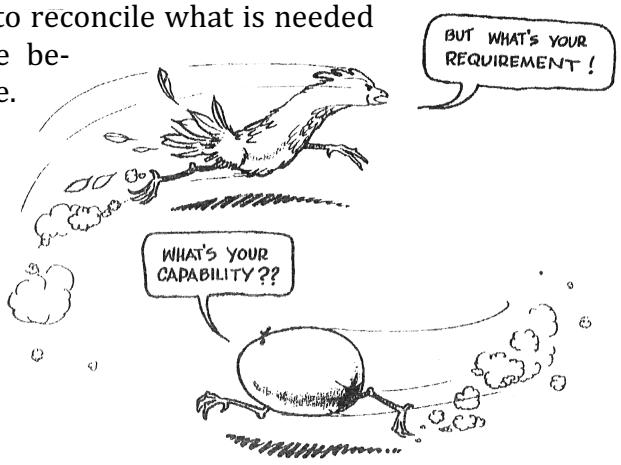
View hierarchies are not common within other architecture frameworks, but they are well suited to projects that concurrently span multiple levels of development, as is typical of flight projects within JPL and across aerospace. Where plausible, the flatter organization suggested by other frameworks can be emulated by imposing levels on the View hierarchy and then strictly layering architecture development. In principle though, it is best to avoid any a priori imposition of such structure. Instead, defining an appropriate hierarchy is treated within this methodology as part of the process, addressed early but allowed to stabilize with the rest of the architecture.

2.1.1 Stakeholders and Concerns: At the top of this view hierarchy are **stakeholders**. In NASA projects, these would include the project sponsor with whom Level 1 requirements are negotiated. But several other institutional, governmental, commercial, or other stakeholders are likely to need formal consideration as well, so it's important to give each their due. These include internal stakeholders with interests in project priorities, guidance for the resolution of competing characteristics, staffing, facilities, and relevance to business strategy.

Specialized views are defined in this architecture framework for characterizing the nature and authority of each identified stakeholder, and for establishing the manner of architectural engagement that each requires. And because a key tenet of architecting is to help stakeholders express their objectives, constraints, or anxieties—to assure them that they are heard and understood—specialized views are also defined to capture **concerns** and to show how success on their behalf is to be measured in verifiable **success criteria**. These are taken as the **originating requirements** on the enterprise. The specialized views in this set are analogous to the routine plans and directives that commission project developments.

2.1.2 Principles: Sharing the top of the view hierarchy are **principles**, as asserted by the architects themselves. While not given the same stature as success criteria, they may nonetheless provide the deciding criteria for difficult choices. The framework encourages explicit documentation of their origin and rationale, both for their contribution to the selected architecture, and for their sustaining value to subsequent efforts. These are loosely analogous to institutionally mandated "principles", but with the intent of being instructive rather than imperative.

2.1.3 Concept vs. Realization: Within the hierarchy of responding views, the framework defines two distinct types: conceptual and realizational. This dichotomy is sometimes explained in other terms—function vs. implementation, logical vs. physical, requirements vs. capability, and so on—but the basic idea is always the same: to reconcile what is needed with what is achievable. There is no precedence between these types in either rank or importance. Their developments necessarily progress in parallel, in a give-and-take manner seeking balance and compatibility.



From JPL D-8614, Rev B "System Engineering at JPL", October 1993 (original source unknown)

Each **conceptual view** considers the architected system from some point of view (science, telecommunication, system integration, data flow, navigation, planetary protection, mission operations, and so on), establishing an approach within that domain for addressing mission needs, while remaining feasible for implementation within the context of the overall design.

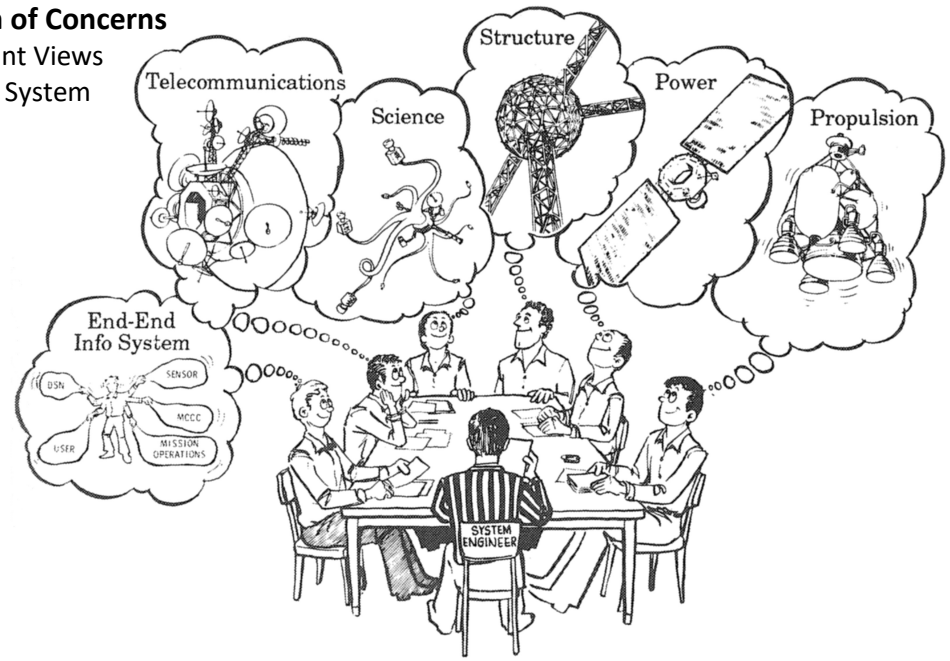
Of special importance among conceptual views is a good separation of concerns[1], such that undiluted attention can be given to the matters at hand. Thus, from their particular overarching perspective, they establish relevant allocations across the suite of products (resource, performance, reliability, etc.), as well as rolled up assessments of system behavior in light of required activities.

Conceptual views are analogous to the many system-spanning documents one expects on any project (e.g., **plans** and **functional requirements** documents, especially at higher levels), each addressing some functional realm, and often provided by specialized project teams. Framework guidance is provided on the overall structure of conceptual views and on their relationships within the view hierarchy, but their specific form has been kept quite flexible.

---

[1] A term probably coined by Edsger W. Dijkstra: "*Let me try to explain to you, what to my taste is characteristic for all intelligent thinking. It is, that one is willing to study in depth an aspect of one's subject matter in isolation for the sake of its own consistency, all the time knowing that one is occupying oneself only with one of the aspects. We know that a program must be correct and we can study it from that viewpoint only; we also know that it should be efficient and we can study its efficiency on another day, so to speak. In another mood we may ask ourselves whether, and if so: why, the program is desirable. But nothing is gained—on the contrary!—by tackling these various aspects simultaneously. It is what I sometimes have called 'the separation of concerns', which, even if not perfectly possible, is yet the only available technique for effective ordering of one's thoughts, that I know of. This is what I mean by 'focusing one's attention upon some aspect': it does not mean ignoring the other aspects, it is just doing justice to the fact that from this aspect's point of view, the other is irrelevant. It is being one- and multiple-track minded simultaneously.*" [20]

Separation of Concerns
different Views
same System



From JPL 89-24, "The Voyager Neptune Travel Guide", Charles Kohlhase editor, June 1, 1989 (artist Phil Gwinn)

Separately, each **realizational view** considers the convergence of diverse concepts on a particular component of the target system: whether an identified product (or integration of products) within the implemented system, or on some part of the system context (e.g., an environment, a service, or some external interfacing system). In this manner, realizational views gather and attempt to reconcile the expectations that conceptual views assert, as applied to a specific item, while counter-asserting interdependencies from practical implementation that conceptual views must honor.

For a product, the resulting reconciliation of these constraints is analogous to some combination of a **requirement** or **specification** document and a **design description** document. For an environment, it would be analogous to the contextual description in an **environment requirements** document. For a service, it would be analogous to a **service specification** (e.g., a DSN Service Agreement). And so on. Guidance on structure and content is comparable to that for conceptual views.

The conventional notion of a "subsystem" typically straddles the conceptual-realizational line, with subsystem teams contributing both conceptually to the system architecture, as well as realizationally to the definition of products. Proper usage of this architecture framework requires these roles to be resolved into their separate perspectives, where subsystem as a conceptual discipline is distinct from subsystem as a realizational product.

For instance:

> Conceptually, telecommunication is broadly engaged with radio signals and associated functions and mediums, trajectory and associated geometry, data and associated capacity and latency demands, etc.
>
> Realizationally, telecommunication is bound to antennas, waveguides, transponders, and related items that must be delivered, and to the accommodation of their electrical, thermal, structural, pointing, commanding, and other dependencies.

2.1.4 Auxiliary Views: Rounding out the set of specialized views are **models**, **scenarios**, **analyses**, and **trades**. Each is defined with the express purpose of encouraging more complete coverage of topics already covered within conceptual or realizational views. In that way, they may be considered as supplemental to those views, in the same way one might regard a document appendix that expands upon some idea addressed tangentially within the narrative, but too complex for full description in that setting.

Within this set, models address items such as simulations, test results, or other sources that purport to describe essential aspects of the architected system or its context. Scenarios address a similar need for the usage of that system, including any other dynamic circumstance relevant to its successful deployment. Analyses address the use of models, scenarios, tools, and other contributions to the assessment of architectural features and capabilities. And trades address the extent, criteria, and review of key architecture choices. In each case, the complexity of the model, scenario, analysis, or trade is expected to govern the scope of its rendering within the framework. Guidance is provided to aid the selection of content, but otherwise, the organization of these specialized views is flexible.

2.2 Composition

Within views, a more detailed framework structure is needed to enable formal expression of the assertions they make. Fortunately, these can all be expressed within a small set of ideas.

2.2.1 Elements: The first of these is composition, wherein the division of a system into parts is defined. Each part is called an **element**, and each separate part of that element is an element, and so on. The most basic of assertions is therefore to declare the required composition of an element into its parts (*aka* sub-elements).

Any given composition of a system is certainly not unique. Each conceptual perspective suggests a different partitioning, whether into thermal zones, fault containment regions, data system nodes, software modules, or gravitational bodies, etc. Composition can also support programmatic or other non-engineering views, such as the structure of a work breakdown or an organization. Separation of concerns among conceptual views permits the independent expression of these ideas.

2.2.2 Functional Roles: Each element within a composition has a role relative to the other elements. Elements that are destined to be products exist to fulfill some purpose. Otherwise, they wouldn't be needed. In this architecture framework, purpose is asserted as the element's **function**, as established by the conceptual view in which its role in the composition is asserted or in the conceptual scenario in which functionality is invoked. An element's function is relevant only within this conceptual context.

2.2.3 Relationships: As parts of a larger system, the elements of a composition must interact or relate in prescribed ways, often in service of the function of the composite element. Elements can also interact in incidental ways that are germane to the architecture, nonetheless. Each such interaction is asserted within the framework structure as a **relationship**. Engineered interfaces

would be included among relationships (e.g., electrical, structural, informational, etc.), but other relevant associations would also be captured (e.g., related to interference, configuration, fault propagation, etc.).

Together, elements and their functions and relationships are clearly analogous to the information typically found in functional block diagrams. As formally modeled items in the framework though, they become relatable entities that are accessible by tools for analysis and other uses. This must be approached carefully. Composition is surprisingly subtle idea [2], addressing a variety of notions regarding the association between parts and whole, and among the parts themselves.

This architecture framework accommodates such issues in part by separation of concerns, but also by classifying elements according to their conceptual or realizational origination. In this manner, potential issues of transitivity can be avoided. Relationships address associations that are non-compositional, and additional associations are defined within the framework for particular dependencies among framework categories. Together, these provide the breadth needed for expressive representations of system composition.

2.3 Commitments

The definition of any element or relationship is accomplished by the declaration of the **properties** it possesses that are relevant to the architecture. Whether simply its name, or more importantly, its physical, geometric, or informational characteristics, properties are the only distinguishing features of an element or relationship, beyond its associations within a composition. Therefore, literally everything that might ever need to be said about them can be expressed in terms of properties and composition. Asserting the existence of these properties is analogous to labeling the columns of a table. These too become relatable entities for external uses.

2.3.1 Properties and Constraints: Properties have values, many varying over time or related to other properties in particular ways. What we know or expect or desire about these values is asserted in **constraints**. Some constraints will simply assign a value (or range of values), while others might be as complex as the relations of an elaborate **behavior** model (i.e., equations and so on).

> Note that named constants are not element or relationship properties under framework definitions. To permit otherwise would nearly always reflect an abuse of notation, where an explicit constraint should be expressed instead.
>
> For instance, it would be inappropriate to say that *maximum_expected_IMU_mass* is a property of an IMU, since this value might change, even though the IMU's actual mass does not. Moreover, the value of *maximum_expected_IMU_mass* is relevant to the IMU only if it participates in a constraint with the IMU's actual mass, which *is* a property of the IMU. Thus, *maximum_expected_IMU_mass* is the name for a value in a constraint on a property, but it is not a property itself.
>
> The framework addresses such named values and the constraints that include them by expressing constraints explicitly (not merely suggesting them in names), and by associating them with the views that assert them rather than with elements or relationships.

A constraint can involve any number of properties, and any given property can participate in several constraints. Therefore, for an architecture to be realizable, properties must be addressed collectively, with all constraints simultaneously satisfiable. With allowances for variation,

determination of relative sensitivities, allocations that permit decoupling, provisions for quantity kind, and so on, something far better than point design assessment is needed. Therefore, the framework encourages representation of constraints in full mathematical expressiveness.

In this architecture framework, constraints are treated as a specialized aspect of a property. For example, a parametric dependency among the elements in a composition would be taken as a property of the composition (as, for instance, it being a "property" of rigid bodies that their parts all move in the same way). Nonetheless, constraints and the values they constrain are held as distinct entities.

2.3.2 Requirements: Each constraint has a provenance that establishes its stature within the architecture. For instance, any constraint asserting a desired characteristic of a product would belong to the conceptual view that makes this assertion, typically in support of some assigned function. Such constraints are analogous to **requirements** in conventional treatments. Indeed, within this architecture framework, this is precisely how requirements are defined ("products" in turn defined by relationships with suppliers). When a constraint expresses an estimated or measured value of a product, it typically belongs to the realizational view that makes this assertion. When a constraint describes intrinsic physical behavior, it belongs to the element or composition that encompasses the scope of that behavior; and so on. Assignment of provenance is a feature of this framework.

2.3.3 Prototypes: Allowance is made in this architecture framework for the concise description of different elements that share a common set of properties (e.g., all hardware has the property, mass). This is accomplished via the definition of **prototype** elements. Any element asserted to be a copy (i.e., instance) of the prototype acquires all of its properties, including relevant constraints (appropriately translated to the instance properties). Relationships among copied elements similarly acquire properties and constraints of relationships among corresponding prototypes. This provides an easy way to levy requirements on *types* of things, the effect being that each instance of that type *also* acquires this constraint, but in a form automatically tailored to that instance.

Both conceptual and realizational prototype elements are allowed, as determined by where they are defined. A conceptual prototype would typically represent a general type of element, serving as a template for different variants, while a realizational prototype would typically specify a particular component that occurs in multiple, interchangeable instances within a system.

2.4 Mapping Concepts to Reality

The convergence of conceptual notions in realizational entities is also accomplished via an association between elements. In particular, every **C**onceptually defined element must ultimately be bound to some **R**ealizational element. That is, each notion must be made real. This is usually a many-to-one mapping, such that a real item might be viewed conceptually as (for instance) a power load, a source of microphonics, a fault containment region, a source of telemetry, and so on. A dozen or more such **C→R** associations would be common.

By these associations, whatever properties, constraints, and relationships are noted for a conceptual element would apply by inheritance to its realizations. In this way, the realization of concep-

tual elements is similar to the copying of prototype elements. However, conceptual compositions may have multiplicity constraints on membership that can be enumerated only via **C→R** associations. These associations also help to address the different ways in which distinct conceptual compositions can overlay the same realizational composition, making them a useful addition to the framework.

Collectively, it is the assertion of compositions, constraints, and so on within a view that become the defining context for hierarchically dependent views. That is, views "see" what is above them hierarchically, but not what is below them. Flow-down is generally established by this means, but it may then be further targeted, such that particular dependent views can be singled out for further elaboration of an assertion. A conceptual view might, for instance, divide responsibility for downlink telecommunications among different elements, one of which has the role of pointing an antenna. Any associated pointing constraint would be directed to whichever dependent view addressed pointing, where a unified approach to all pointing needs would be addressed.

Traceability from view to view is augmented by the net of associations among constraints through their shared properties. This is closely analogous to connections from requirement to requirement in a conventional approach, but with the connections being formally and verifiably determined by the constraints themselves, and with their scope being wider than requirements alone (e.g., traceability to assumptions about the environment).

In this light, it's vital to note that the C→R association is accessible to conceptual views, such that any constraints originating at the realizational level on properties defined at the conceptual level are within the purview of the defining conceptual view. This enables the assessment of closure within an architecture, whereby the realization of a conceptual approach can be examined or even further constrained in order to declare compliance or to push back for relief.

This give and take highlights a vital aspect of the architecture framework wherein conceptual and realizational views are properly viewed as peers, neither being "below" or subordinate to the other. They share a set of properties, constraints, and so on, as defined via **C→R** associations, through which their mutual compatibility can be established. What flows from the conceptual side are the successively refined ideas that establish system functionality. What flows from the realizational side are the practicalities of implementation, integration, and deployment. Architecture mediates their association.

2.5 Viewpoints

Viewpoints are the final category defined by this architecture framework. They fill a role somewhat related to principles, but dedicated to the *practice* of defining an architecture, rather than to the architecture itself. The intent of each viewpoint is to set standards, conventions, or other criteria that would be expected from a well-crafted view in a particular topic area or discipline. In addition to defined abstractions intended for use in developing descriptions from this point of view, viewpoints could include accepted methods, sanctioned data sources, useful reference material, mandated reviews, or supported formats, notation, and tools.

Like principles, viewpoints would be expected to provide sustaining value to subsequent efforts, improving and evolving from one application or project to the next. Consequently, a viewpoint is the sort of institutional asset that would normally be associated with some discipline group having responsibility for its development and evolution (comparable to institutional procedures and guidelines). In this regard, viewpoints are considered authoritative, but they needn't be

peremptory. Their aim is to guide and support disciplined developments, and to establish common practice and understanding.

2.6 Summary

The framework outlined above is a collection of 14 entity types, each intended for the expression of information in different **categories**. For each such category, information is expressed in **records**.

> For example, each View would be one record, and all View records collectively comprise the contents of the View category.
>
> ***Note the capitalization of "View" above. From here through the end of this document, wherever a category is mentioned, it will be capitalized in order to emphasize the formality of its place within the architecture framework.***

The records in a given category have a minimally defined structure consisting of a small number of **fields**. These include name and description fields, plus added information fields appropriate to the category.

Records may be connected to one another in ordered pairs via **associations**. Only a subset of category pairs support associations, but where an association is permitted, it simply links two records of the indicated categories.

> The details of framework records, fields, associations, rules, etc. are documented, with examples and further guidance, in **JPL D-55628** Architecture Framework Definition [21].

> For example, a particular View may address a particular Concern. This pair is associated via an *addresses* association.

There are a few rules about the multiplicity of associations. These determine which combinations of associations are permitted to appear in the same architecture description.

> For example, no Property can belong to more than one item. That is, any given Property can appear in only one *belongs to* association.

2.6.1 Categories

The complete set of categories is as follows:

Stakeholder	A Stakeholder record describes one of the parties to whom a project is obligated regarding performance or outcomes. These would typically include the sponsor for a project plus various institutional or governmental entities.
Concern	A Concern record describes an issue of importance or interest to a Stakeholder. Concerns are the source of originating constraints on a system. For NASA projects, these would include Level 1 requirements. Others would typically involve institutional or legal mandates, partnering agreements, or comparable
Viewpoint	A Viewpoint record provides guidance for development and content of Views addressing a particular topic or discipline. A Viewpoint can establish representation and analysis conventions; it can provide vetted sources of data and authoritative reference material; it can recommend areas that need attention, processes for development, and so on. Viewpoints would typically be organized in a manner analogous to the conventional partitioning across procedure or

policy documents. They would be reusable across projects.

View A View record describes a particular aspect of an architecture from some point of view. A View may directly address some Concern; it may elaborate some particular aspect identified more generally in another View; or it may serve to reconcile overlapping interests. Views would typically be organized in a manner analogous to the conventional partitioning across design or specification documents.

Trade A Trade record describes the basis for a significant decision made by a project. Trades would typically address technical choices among design alternatives, but they could also address programmatic choices. Each Trade would provide information needed to understand the options considered and the criteria by which the choice is made.

Model A Model record describes our understanding of some aspect of a system or its mission, and the manner in which this understanding is to be applied in analyses, simulations or other applications. A Model would typically be created to explain some complicated aspect of a system or mission that would benefit from dedicated treatment outside of Views and Viewpoints.

Analysis An Analysis record describes a calculation, test, or other assessment that is performed in the evaluation of the architecture. An Analysis might be a one-time case, for instance to support a Trade, or it might be one that is repeated routinely as part of the iterative process for arriving at a closed architecture.

Element An Element record describes one of the parts of a system. These would typically be engineered components, but they could also include external services used by the system, relevant objects in the environment of the system, and other systems with which the system must interoperate. In addition, Elements can be defined that describe types of Elements.

Relationship A Relationship record describes one way in which certain Elements of the System interact with one another. Relationships will typically describe functional interdependencies, sharing of resources, sources of interference or contamination, and so on.

Function A Function record describes the purpose or role of an engineered Element in an architecture. This is typically an expression of the division of responsibilities observed in an architecture.

Property A Property record defines a particular attribute, quality, or characteristic of something in the system, including any definitive constraints. A Property can describe physical, informational, procedural, temporal, and other aspects of Elements, Relationships, Scenarios, or other items. Constraint may express intentions that lead to requirements on project products, or they may express assertions regarding what is known about other items of interest to the project.

Scenario A Scenario record describes a situation that a system could encounter and the manner in which it is intended or expected to unfold over time. A Scenario would typically address a particular planned activity, before, during, or after a mission, considering how it might be done and whether objectives for it can be met. A Scenario could address nominal behavior, or it might instead explore the implications of faults, discoveries, or other unplanned events. Scenarios

become the driving cases for evaluation of the system.

Principle — A Principle record captures some notion that is considered fundamental to good architecture. Principles will often be relevant to a particular technical discipline, but general principles may also be identified. Principles should be applicable across all projects.

Requirement — A requirement record is the rendering of a particular constraint on a particular Element in the form of a "shall statement".

2.6.2 Associations

The complete set of associations that are defined among categories (with their associated multiplicities) is as follows:

Each Concern is important to one Stakeholder.
Separate Concerns may be similar, but to ensure that the nuances of each Stakeholder's interest are not lost, each Concern is associated with an appropriate Stakeholder, and not shared other Stakeholders.

Each Viewpoint is prescribed by zero or more Stakeholders.
It is common for Stakeholders to set expectations for the manner in which their Concerns are addressed (documentation, analyses, reviews, etc.). These expectations are captured in Viewpoints associated with the governing Stakeholders.

Each View addresses zero or more Concerns or Views.
Through this acyclic association, all Views are traceable to Concerns, either directly or through the View hierarchy. This traceability is further extended to auxiliary view, as noted below. This hierarchy also governs the scope of Views and auxiliary views, which should not refer to subordinate content.

Each View conforms to zero or more Viewpoints.
This association establishes which Viewpoints assert guidance for each View, and it places Viewpoints within the scope of the Views they guide.

Each View is provided to zero or more Stakeholders.
This association establishes certain Views as subject to Stakeholder review.

Each Trade is reflected in zero or more Views.
The decisions reflected in Views often arise from formally documented Trades, which are traceable via this association. This association places Trades within the scope of the Views they inform.

Each Trade considers zero or more Concerns or Principles.
This association ties the decisions of Trades directly to any Principle or Concern that factors significantly influences their outcome in a direct manner.

Each Model is defined by one View or Viewpoint.
Models are auxiliary views that document complicated aspects of a system that Views and Viewpoints depend upon. This association places Models within the scope of the Views and Viewpoint they support.

Each Analysis is invoked by zero or more Views or Viewpoints.
Analyses are auxiliary views that may either describe a particular investigation that directly complements the exposition in some View, or that prescribe a type or method of analysis that would be expected from some Viewpoint. This association places Analyses within the scope of the Views they support.

Each Element is defined by one Concern, Model, View, or Viewpoint.
Each Element must be defined within a particular context. It is then accessible to any other context that has the defining context in its scope.

Each Element is a copy of zero or more prototype Elements.
Through this association, Elements can be declared to be of a particular kind, as defined via a prototypical Element, in which case they acquire the characteristics of the prototype.

Each Element is a realization of zero or more conceptual Elements.

Conceptually established Elements are given concrete realizations through this association. Realizational Elements can participate in multiple conceptual contexts.

Each Element is a member of zero or more composite Elements.

This association establishes the connection between a composite Element and the sub-Elements that comprise it. In the conceptual realm, this association is at the core of functional decomposition. In the realizational realm, this association is at the core of product breakdown and system integration.

Each Element is a participant in zero or more Scenarios.

Functional dependencies are often established through Scenarios that describe the manner in which Elements interact. This association formally establishes such dependencies.

Each Relationship is defined by one Concern, Model, View, or Viewpoint.

Each Relationship must be defined within a particular context. It is then accessible to any other context that has the defining context in its scope.

Each Relationship is a connection among two or more Elements.

This association establishes the network of Relationships among the Elements in a system.

Each Scenario is defined by one Concern, Model, View, or Viewpoint.

Each Scenario is motivated by a particular context in which the system is applied. It is then accessible to any other context that has the defining context in its scope.

Each Function is assigned to one Element.

In a proper functional decomposition, each Function must be associated uniquely with an Element that fulfills this purpose.

Each Function is in support of one composite Element or one Scenario.

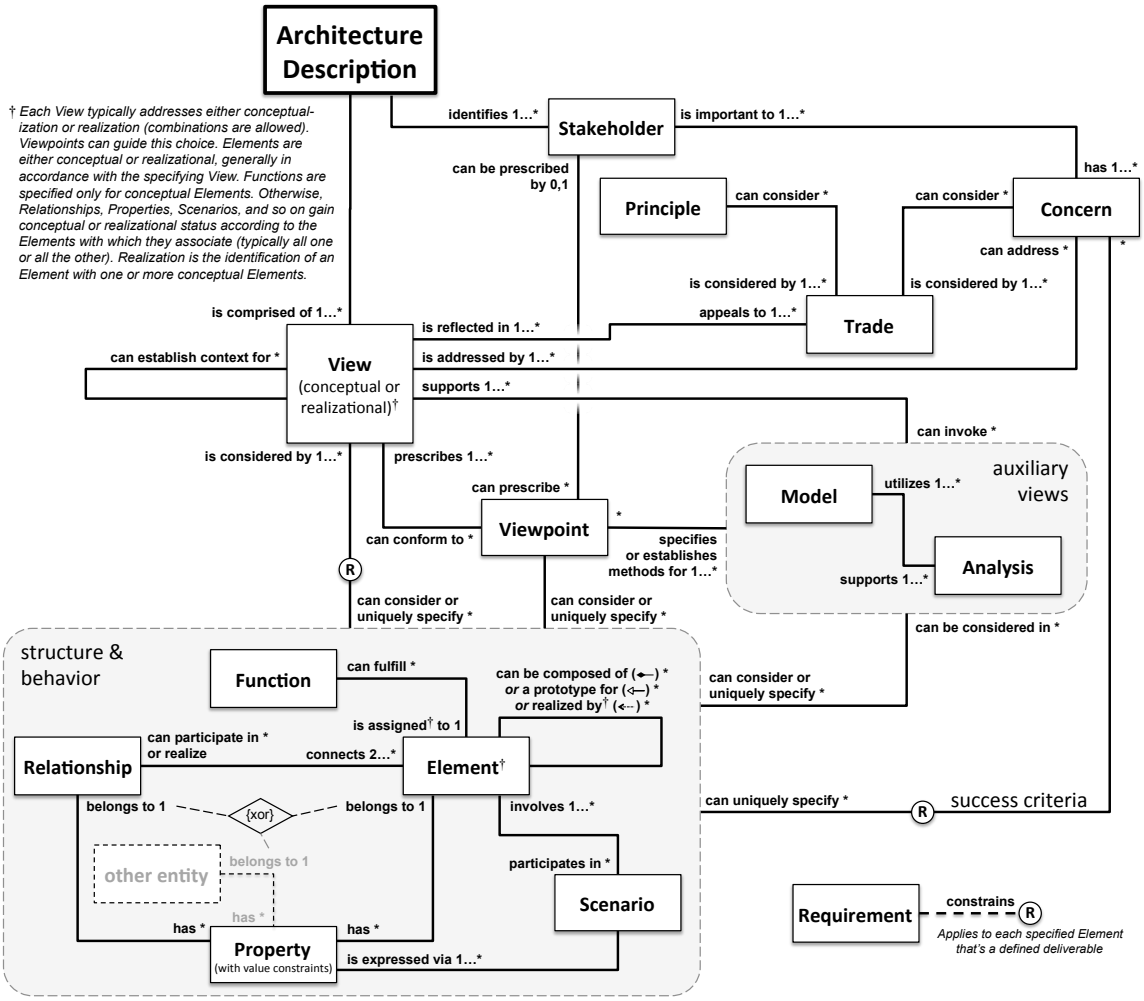
This association establishes the traceability of Functions to the integrated contexts within which they are relevant.

Each Property is possessed by one Analysis, Concern, Element, Model, Relationship, Scenario, View, or Viewpoint.

Every Property needs to be associated either with something that intrinsically possesses a characteristic described by that Property or that asserts a characteristic via that Property to be imposed on something else.

2.6.3 The Framework Diagram

Framework classes and the associations among them, as described above, are summarized in the following diagram.



Key: Each box denotes a class within the framework. The box label is a noun phase naming the class.

A line connecting two boxes denotes a recognized association between two items of the connected classes. The label on the line next to an item is a verb phrase describing the role that the adjacent item plays in the association. Thus, for each association, class-role-class may be read as a sentence.

Roles can be assigned to both items in an association. If a role in one direction is missing, it can typically be inferred from the role in the other direction. Roles typically contain multiplicity information describing how many items of a class can participate in an association. The shorthand notation for this is either a non-negative integer, a comma-separated list of non-negative integers, or a range denoted by two ellipsis-separated, non-negative integers. An asterisk (*) indicates the absence of a bounding constraint. If multiplicity information is missing, '1' is implied.

■

3 Viewpoints

As generally considered in architecting literature (e.g., [3], [4], [5], [6], [7], [8], [9]), Viewpoints articulate Stakeholder Concerns in a practicable form and they establish expectations for the Views that respond to these Concerns. In the framework set forth here, Concerns are kept separate from Viewpoints, but both are related to Stakeholders, such that their mutual contributions to Views are handled appropriately. This permits an association between the Viewpoints described here and kinds of Concerns that generally motivate these Viewpoints. The Viewpoint descriptions below are consequently organized by recurring Concern themes.

The Viewpoints have been organized in two categories: conceptual and realizational. It should be understood, however, that **some Viewpoints will necessarily stipulate both conceptual and realizational View content**. This is generally appropriate where conceptual issues are narrow and apply within a similarly limited realizational scope, or where a conceptual-realizational dichotomy is not useful.

3.1 Conceptual

Conceptual Views generally address both the **approach** to be taken in response to an issue and an **assessment** of the closure of this approach (Is the defined approach actually realizable? Can constraints be met? How well? Is there sufficient tolerance to variation to protect development? Etc.). As necessary, they add any remaining details or constraints needed to affirm this, where there is dependence on design choices made elsewhere. Analyses are expected to figure prominently in conceptual Views, particularly in regard to implications of the total architecture on that aspect of the architecture to which the View is dedicated.

The delineation between an approach and its assessment in a View is seldom distinct or precise. These aspects are complementary though, in the sense that an approach emphasizes separation of concerns, giving weight to some point of view such that its interests are satisfied, whereas an assessment emphasizes harmony within an integrated architecture, showing that this point of view can be accommodated among other coupled interests without losing the integrity of a well-considered approach.

A common aspect of assessment analyses is to consider specific cases arising from realization that could be addressed only in generalities by the approach. In a typical conceptual View, the approach might define a pattern of composition or interaction, declaring general constraints that would apply for all cases. Assessment would then inspect each realized instance in turn, confirming that constraints are met in each case (and barring exceptions, as discussed below). Where this confirmation is contingent on additional information that has, as yet, remained open, assessment might further constrain such contributing factors, as necessary to ensure closure. For instance, an approach for managing power would routinely define an allocation method for power loads, but particular allocations can only be addressed in assessment, once all the realized loads are identified.

Frequently, the information contributed by a realizational instance is its own particular set of constraints, to be combined and reconciled with those of the approach, such that the contribu-

tion of the conceptual View is to provide the pattern of implementation by which such constraints can be met, while realizational Views provide the particulars of one instance of that pattern, which can then be specifically addressed. For instance, an approach for addressing the mechanical configuration of a system would routinely include the definition of different types of shapes, among which various geometric relationships, according to type, would be constrained[2] —something that can only be addressed fully in assessment, once all the shapes and their relationships are identified.

Similar situations arise where the variant in a pattern is a type of activity, rather than a type of thing. In these cases, assessments would examine each specialized scenario in turn, something that can only be addressed once all the activities of that type are identified as integrated Scenarios in realizational Views.

Each of these examples describes analyses that are familiar and routine. For some, analyses are most appropriate relative to constraints (e.g., requirements) rather than design, because an architecture should close for any implementation that complies with constraints. These analyses would be updated as constraints mature, but they would be expected to hold over time once constraints are settled.

For a few other analyses, there are general expectations during development that projected performance for the current design is reported regularly as it evolves. Such cases typically address tolerance to variation (e.g., margins), where routine reassessment is necessary in order to safeguard against erosion of this tolerance. These reporting aspects of assessments should be expected to change over time.

Such reports are often in tabular form, because they iterate across a set of related instances or cases, as described above. However, there is no set expectation. Wherever such a report is warranted, in whatever form it appears, the appropriate architectural home for it is in some conceptual View's assessment, either directly or by clear, controlled reference (e.g., to some auxiliary view).

In all cases, the ambition of each assessment is to provide a validation of the approach, where the analyses provided and the mutual acceptance of flow-down constraints in the realized architecture combine to give assurance that objectives are achievable.

There may nonetheless be cases that take exception to a prescribed approach, whether by specially allowed departures or (in problematic cases) by formal waiver. Where exceptions are negotiated, these should be identified and justified as part of the conceptual View. On the other hand, waivers are (by definition) acknowledge escapes from a considered architectural approach, so they are accordingly justified and approved elsewhere by extraordinary means. In either case though, analyses, allocations, and so forth should make explicit reference to exceptions and consider them in the reported results.

Following are descriptions for particular conceptual Viewpoints. General guidance above applies, but particular Viewpoints would be intended provide more directed advice for what might be

[2] Solids can't intersect, fields of view and articulation volumes can't be blocked, everything must be attached to something, items must fit within assigned enclosures, etc.

included in governed Views, and to provide additional supporting information, representation and analysis conventions, expectations for resulting products, and so on.

In most cases, what follows are only brief descriptions of the intended Viewpoints. However, in a few selected cases, referenced below, more elaborate descriptions have been provided in Appendices that show in more detail what such Viewpoints might be expected to provide.

3.1.1 Mission Concept: As necessary, Views governed by a "mission concept" Viewpoint can serve as bridges from Stakeholder Concerns, to the various Views that address them. In this role, they can define the top-level structure of project products, initiating functional decomposition of the project system.

Mission concept Views can also provide a direct assessment of the project system architecture relative to success criteria. This could involve independent summarizations of assessments from subordinate Views, or a compilation of evidence from multiple Views that supports an integrated argument for compliance, including robust margins against success criteria. Part of this assessment should address the top-level realizational breakdown of the project system, supporting an assertion of balanced and thorough treatment of all project system Elements.

3.1.2 Science: Views governed by a "science" Viewpoint are appropriate for all missions with science objectives (most of those performed by JPL). Besides addressing the technical aspects of collecting science data, science Views also consider the data's suitability to the science objectives of the mission and means by which scientific results are produced and communicated.

Objectives, Investigations, and Measurements

Conceptual Views governed by this Viewpoint establish the overall pattern of science activity for a mission, explaining generally what science data sets must be acquired to meet science objectives (typically Level 1 requirements), and how this is accomplished via a combination of mission and system features.

A common feature of governed Views will be a **science traceability matrix** for the project (or something equivalent), laying out the intended investigations and an approach to their accomplishment. By this means, Views should address broadly how well the design meets its intent, as rolled up from lower-level conceptual Views and from realizational capabilities. For example, this would be a reasonable home…

- for defining each of measurement types needed to fulfill the approach,
- for tracing effectiveness of the defined instrument and mission designs in addressing required science investigations, including a variety of coverage statistics or maps, and with allowance for failures or other threats,
- for presenting analyses that show metrics relating capabilities and plans to progress over time in reaching each science success criterion, and
- for an assessment of any compromise situations where the assertion of science priorities was needed in order in order to resolve competing interests.

Other analyses should also be considered where they assess the overall capability of the mission to accommodate science interests, such as flexibility for continued mission refinement, adapta-

bility and operational responsiveness to scientific discovery, potential for an extended mission, and so on.

Observation Strategy, Opportunities, and Planning

Conceptual Views governed by this Viewpoint address mission design aspects of the observation strategy, laying out constraints on trajectory and pointing that determine coverage, viewing geometry and lighting, and other observational factors (e.g., speed or environment) that are compelled by science objectives, as broadly determined by the approach defined in Objectives and Investigations Views.

Checkout and calibration opportunities should also be addressed. These are analogous to observing opportunities, but relative to supporting data needed to understand and adjust measurement collection performance, rather than to the science measurements themselves. This includes the identification of activities, targets, times, and so on that must be included in mission plans in order to provide this complementary data set.

These Views are the primary source of reconciled science constraints on a mission's trajectory. Given a variety of science interests, governed Views should reflect a compromise of competing factors that must be reconciled across the instrument suite and within engineering restrictions. As this evolves, Views should address the scientific adequacy of the current mission design trajectory and associated observation plans.

Where an approach cannot assert a priori what science opportunities may be specifically, Views must instead inspect the set of all potential opportunities identified in the current mission plan. Therefore, analyses such as coverage assessments, a summarized under Objectives and Investigations Views, would be addressed in detail in Observation Planning Views.

Assessment of checkout and calibration opportunities should similarly assess the current mission design trajectory and associated checkout and calibration plans. Details of supporting analyses and addition of supporting refinements would be included.

Potential supporting refinements, as driven by particular measurement capabilities, operational considerations, or other issues can also be added.

Sensing and Data Acquisition

Conceptual Views governed by this Viewpoint address a related collection of observation capabilities, as needed to acquire particular science data sets. Each is usually assigned to one instrument (or a small subset of the instruments) with particular performance needs, accompanied by additional conditions on the mission plan and supporting spacecraft capabilities, such that each data set has the necessary measurement quality and coverage, capacity for correlation with other data sets, acceptable size relative to storage and downlink capabilities, etc.

Topics can cover items such as 1) margin of the realizationally defined system (instruments and their accommodation, mission plans, etc.), relative to measurement requirements, 2) implications of performance limitations, operational restrictions, or other factors identified in the mission and system designs, especially for which further constraints may be needed in order to ensure satisfactory results, or any additional capabilities not strictly required, but available, should opportunities arise. To the extent that measurement-specific constraints must be im-

posed on the trajectory, on delivery and pointing performance, or on other capabilities, these should be directed for reconciliation by other Views.

Science Data Management

Conceptual Views governed by this Viewpoint address the handling of scientific data generated during a mission. This would typically be confined to operations occurring after downlink, but it could begin with its initial production on the flight system, if there are scientific considerations in the methods by which raw data is subsequently stored, processed, or transported before being resident in ground systems. For instance, in a highly autonomous system, early stages of science data management might occur onboard.

Science data management begins with archiving and curation, in order to track and characterize everything that is collected. Views would address how and where this is to be performed, and how other sources of relevant information are integrated (e.g., navigation and pointing, instrument status, timing…). To be considered as well are further data products developed to present more refined renderings of the data and contribute to the extended science data archive.

Views should also address the distribution of science data, in all its forms, considering accessibility, searchability, or other performance factors that govern it availability. This typically involves rules of precedence among science groups established during the course of development.

Depending on the nature of the mission and its scientific objectives, there may also be expectations regarding the scientific analysis and interpretation of data, with expectations governing the content, manner, and timing of reported results, including to the broader public audience.

3.1.3 Engineering

Engineering: Views governed by an "engineering" Viewpoint define the plans and systems that must be engineered in order to satisfy mission objectives. Capabilities of both flight and ground (and other systems, as appropriate) are covered. Regardless of whether a combined flight-ground discussion is called for, or flight or ground dominates in a particular area, appropriate coverage for each is explored within the Viewpoint.

Mission

Mission Design — *To be provided*

Trajectory and Integrated Targeting — Conceptual Views governed by this Viewpoint address the numerous competing demands on flight system trajectory, describing how these demands can be efficiently and safely met within practical physical capabilities of the launch, flight, telecommunication, and other systems. These Views establish an overall approach for a mission, which is then elaborated in detail, considering performance needs regarding geometric relationships (location, orientation, etc.) that are important to a variety of project system functions. These relationships exist primarily between features of the flight system and external objects of interest, where quality of knowledge or control is important; and because of relative motion, time is also a factor.

Views should assess broadly how well the overall system and mission design meets the demands upon it. This would be a reasonable home for summarizing margin relative to driving parameters of the mission design, especially with respect to one or more design reference missions, which

should be defined in a realizational View. Views should also consider and reconcile needs, as manifest in a set of related realizational Scenarios, allocating the resulting consolidated set among navigation, pointing, delta-v, and time management capabilities within various Elements of the project system. These are shared assets, as exposed in realization for both project system and external Elements, so analyses must demonstrate that the defined system meets mission needs. These results would comprise the usual suite of navigation, pointing, and time analyses and their associated top-level allocation budgets, potentially imposing additional constraints in order to ensure that each Scenario can be handled.

Launch Services and Performance — *To be provided*

Telecommunications Strategy and Opportunities — Conceptual Views governed by this Viewpoint address telecommunication and radiometric needs, accommodating constraints imposed by physical and operational limitations (e.g., data, trajectory, power, pointing, etc.) as well as predefined infrastructure (e.g., ground stations, orbital relays, etc.), in order to arrive at a strategy that is well suited to the mission.

Views should assess each of the demands on this capability that arise from operational Scenarios defined for data management and transport, navigation, gravity science, and perhaps others, which are then manifest in a set of related realizational Scenarios. These will likely involve shared assets, as exposed in realization for both project system and external Elements, so analyses must demonstrate that allocations (capability, time, bands, etc.) from the defined systems meets mission needs. The results would comprise the usual suite of link analyses and their associated allocation budgets, as prescribed by the Viewpoint, potentially imposing additional constraints in order to ensure that each Scenario can be handled.

Data Management and Transport — Conceptual Views governed by this Viewpoint address the end-to-end flow of data within the project system. Such data arises from many different sources, both flight and ground, and needs to move to many different destinations, also both flight and ground. There are connections and intermediaries along the way, which must meet performance and integrity constraints that vary according to data and circumstance. Moreover, these assets must be shared among competing demands. The approach described in these Views lays out the general methods, principles, organizing structure, and capabilities by which this array of needs are to be addressed.

View assessments should identify and address each of the variants that arise. For example, this would be a reasonable home for analyses of science data throughput, latency, and related performance measures—each relative to link rates, onboard data storage capacity, downlink opportunities, and so on. Similar analyses would be needed to address various uplink capabilities (for nominal and contingency situations), secure data transport, etc. These variants are determined by considering all realized data source-destination paths among project system Elements, all realized Scenarios of data movement (including overlapping needs), and the technical capabilities assigned to all realized Elements within the end-to-end data flow.

Time Management — Conceptual Views governed by this Viewpoint address the performance of clocks used throughout the system, defining the means by which needed accuracy and stability are accomplished. There are many clocks, both flight and ground, each with different performance. Therefore, some correspondence among them must be established in order to ensure adequate coordination and correlation of activities in different parts of the system, in accordance

with a variety of needs. Views lay out the general methods, principles, organizing structure, and capabilities by which these needs are to be addressed.

Views should gather and address each of the variants that drive clock performance and time management functions. These variants are determined by considering all realized clocks and clock correlation features among project system Elements, and all realized Scenarios of time usage and coordination.

Operations

Autonomy Strategy — Conceptual Views governed by this Viewpoint address the division of responsibilities between mission operators and flight system autonomy, the latter referring to planning and reactive control functions allocated to the flight system without "ground in the loop". The functions involved would include planning and execution of flight system activities for both routine and anomalous situations, as well as all associated ground support capability.

For some functions, this operational split is generally not static, varying by both circumstance and experience. It is also not an all-or-nothing choice, comprising a mix of responsibilities across functions. Therefore, the approach described in these Views enumerates the spectrum of mission Scenarios that must be differently accommodated and describes for each the selected mix of functions and the criteria by which the allocation is made, including any degrees of freedom in this allocation that must be preserved for subsequent discretion by operators during the mission. Also addressed is the means by which operators can adequately assess flight system behavior and circumstances, and then exercise their discretion in authorizing flight system autonomy.

Having addressed how responsibilities are divided, further guidance for the flight system is provided in the Planning and **Execution** Viewpoint, and for mission operators in the **Mission Operations** and **Mission Planning and Activity Generation** Viewpoints.

Mission Operations — Conceptual Views governed by this Viewpoint address the spectrum of capabilities and activities needed to successfully operate a system during its mission. This includes a definition of plans, methods, teams, and supporting tools and procedures used by operators, as well as direction regarding the operational attributes of deployed systems and plans for their usage. As the culmination of long preparation, the resulting mission operations approach must reflect an integration of these factors that works smoothly, reliably, and efficiently in the presence of unfolding changes over the course of the mission. Views should establish a general approach for these issues.

Views should also consider the patterns of planning, team organization, monitoring and control, analysis, and so on that have been defined in the Approach, addressing each of their instances in realization for adherence to the approach and for further elaboration of detail, as needed for particular cases. For example, if there is a general pattern for flight system resource management, then for each instance of a managed resource that is realized among flight system Elements, an assessment should ensure that the necessary operational components of resource management for that particular item are in place, levying additional constraints as necessary to ensure satisfactory performance (e.g., constraints on assessment rate or margin). Similarly, most integrated Scenarios should be identified as realizational instances of activities to be conducted by operations, and then assessed according to the established approach, and so on.

Scenarios & Operating Behavior — *To be provided*

Flight System Disciplines

Instruments — *To be provided*

Planning and Execution — *To be provided*

Physical Structure and Configuration — *To be provided*

Mass and Inertial Properties

See **Detailed Viewpoint Example, Appendix A.1**

Attitude, Articulation, Maneuvering, and Deployment — *To be provided*

Propulsion — *To be provided*

Energy and Temperature — *To be provided*

Radio — *To be provided*

Telemetry — *To be provided*

Mission Robustness, Fault Tolerance, & Redundancy — *To be provided*

Electrical Design (including interference management) — *To be provided*

Computing, Networking, and Data Storage — *To be provided*

Software — *To be provided*

Materials, Processes, & Contamination Control — *To be provided*

Quality and Reliability — *To be provided*

Environmental Compatibility — Conceptual Views governed by this Viewpoint identify environmental issues that can arise in the planned mission and describe the various means by which the project will characterize and address each aspect of this environment. Environments comprise both external, artificial or natural phenomena as well as internally induced, physical side effects from one system part onto another. In general, constraints may be imposed on the mission or flight system designs in order to limit the severity of these environments. Uncertainty is addressed by asserting conservative bounds within which designs must be tolerant. The selected approach should define the nature of these provisions and should quantify bounds where a priori criteria can be asserted.

Views should complete the quantification of environmental constraints, as determined by information available from the realizational design. For example, a design reference mission is a realizational compilation of limiting mission Scenarios that is purposefully intended to establish engineering bounds on driving aspects of the mission plan. All constraints appealing to the design reference mission for definition should derive from View considerations. Similarly, where

specific reference is made to parts or properties that derive from system design, these Views are the appropriate home for associated constraints.

Ground System Disciplines

Mission Planning and Activity Generation — *To be provided*

Engineering Performance Analysis — *To be provided*

Data Handling, Display, and Archiving — *To be provided*

Tracking and Navigation — *To be provided*

Science System — *To be provided*

Integration & Test

Venues & Integration Flow — *To be provided*

Simulation & Support Capabilities — Conceptual Views governed by this Viewpoint define the set of project-unique support equipment, facilities, and related items that the project must produce in order to enable or assist the integration and test of other flight and ground Elements of the project system. This View describes the types of items that are needed, according to various patterns of support, and provides general criteria for establishing the capabilities and performance of such items. For instance, there might be patterns for enabling test of partial systems during integration, or patterns for closing loops around system functions that can't operate normally in a test environment, or patterns for non-interfering monitors of system interfaces, etc. Similar patterns can be defined for integration and test activities.

Assessments should address each instance of the established patterns, further elaborating any specific needs that arise from the particular details of this case. In most cases, patterns will involve items of the system being integrated or tested, so corresponding elaboration for such cases will define the associated support capabilities needed to enable that process.

System Instrumentation — *To be provided*

Transportation — *To be provided*

Launch Preparation — *To be provided*

3.1.4 Programmatics*:* The purpose of Views governed by "programmatics" Viewpoints is to address the composition and structure of the organizations responsible for developing and operation a system and to define the plans, processes, methods, rules, resources, and so on under which their work is performed. These are necessarily part of an architecture description, because they greatly influence the composition and structure of the architected system.

Development

Project Implementation — *To be provided*

Systems Engineering — Conceptual Views governed by this Viewpoint establish the process for developing and validating the project system architecture, and for asserting architectural guidance throughout implementation and test. As an important part of this, they also address the manner in which architectural choices are narrowed (e.g., through Trades and Analyses), and in which tolerance to variation is maintained across the life cycle, whether through management of margins, preservation of design alternatives, or provisions for operational contingencies, as necessary to address uncertainties. The particulars of these measures are further developed according to principles and oversight explained in these Views.

Views should also summarize the assessment of each aspect of the process established in the approach and elaborated in subordinate Views. These can address a set of related issues for which a tailored aspect of the process has been warranted—typically involving the definition of one or more categories of interest, for which various instances should be identified, tracked, and documented. Different subordinate Views would address different categories.

For each identified case, an up-to-date assessment should be provided of current status, relative to the criteria established in the approach for items in each category. Where criteria should apply across all designs that comply with requirements, assessment should be relative to what is allowed by requirements. Where criteria should apply to the current design (e.g., margins), assessment should be relative to the current design.

System Development — *To be provided*

Risk Management — *To be provided*

Inheritance & New Technology — *To be provided*

Project & System Level Functional V&V — *To be provided*

Software Development and IV&V — *To be provided*

Business

Flight Hardware Logistics — *To be provided*

Acquisition — *To be provided*

Cost & Schedule — *To be provided*

Management Structure — *To be provided*

Work Breakdown

See **Detailed Viewpoint Example, Appendix A.2**

Information and Configuration Management — *To be provided*

Mission Assurance — *To be provided*

Security — *To be provided*

Project Implementation — Conceptual Views governed by this Viewpoint broadly consider the programmatic aspects of developing, implementing, and operating the project system. This is accomplished mainly through processes for managing the variety of efforts conducted by the project team, and through policies that establish overall guidelines and priorities. Where suitable, these may be elaborated in subordinate Views, with focus on crosscutting topics most appropriately considered at this higher level.

Guidance is provided in this Viewpoint for all phases of a project's lifecycle, typically beginning in pre-project development, where initial architecting occurs, and ending in project closeout. Views governed by this Viewpoint would be similar to typical project implementation plans.

Project Reviews — *To be provided*

External Communications — *To be provided*

3.1.5 Public Policy: Views governed by "public policy" Viewpoints address a broad spectrum of Stakeholder interests beyond the immediate objectives of a NASA science project. These can involve effects that a project's actions may have on safety, on future projects, on community support, and so on.

Environmental Safety

To be provided

Planetary Protection

Conceptual Views governed by this Viewpoint lay out a gamut of diverse measures taken during development and operations to successively reduce the likelihood of violating planetary protection criteria to an acceptable level (as driven by success criteria associated with the mission's classification). Each stage in this reduction imposes constraints on the mission and system that must be analyzed. In addition, a variety of assumptions must be posed regarding environment, reliability, biological survival, and other factors that contribute to the cumulative probability.

Assessments should present an analysis of these measures that confirms the viability of the chosen Approach. In addition to evidence validating any supporting assumptions, this analysis would include an accounting of identified realizational Elements that participate in the analysis, along with realizational Scenarios upon which analysis is constructed.

Orbital Debris

Conceptual Views governed by this Viewpoint are typically simple, mainly explaining the project's approach to avoiding the release of material (by various means, accidental or otherwise) that might remain in Earth orbit. This results in general constraints on various Elements of the flight system (e.g., deployable items, pyro actuations, energetic devices, etc.), and on its initial trajectory.

Views should assess the planned trajectory, operational plans, and characteristics of realizational Elements identified as potentially relevant to debris management constraints, showing that the combined hazard is acceptable.

Public Information

To be provided

3.2 Realizational

Realizational Views broadly address the fulfilment or instantiation of conceptual notions in practicable form. Therefore, they necessarily deal with **concrete products**, with their **planned deployments and usage**, and with the actual entities comprising their **operational context**

(e.g., with realized interfaces to other realized components in integrated scenarios). These represent the culmination of conceptual notions in real entities that can be built and utilized to achieve architectural objectives.

The essential purpose of a particular realizational View is to gather and reconcile all relevant assertions about a particular *kind* of deliverable product within the architected system (all instances of which share a common description), or about a particular related item in an external system or environment, or about some meaningful composition of such things. These are the realizational Elements of the architecture.

Separate realizational Views would typically describe distinct kinds of Elements or their integrated compositions. The emphasis on *kind* here is important, because the realizational Element item addressed by a realizational View may be instantiated in multiple delivered units of that kind. Moreover, as part of an architecture description, Views need assert only that which is relevant to the architecture. If units of different designs can satisfy the same constraints, then any of them would be acceptable, and the realizational Element effectively describes them all. Even where only a single specific instance is intended, kind may still be relevant as an expression of the features it shares with others (e.g., Jupiter is a particular planet, but all Solar System planets orbit the Sun).

The character of each realizational Element is established via its identification with various conceptual Elements. A conceptual Element is merely a restrictive representation of some realizational Element (or a class of such Elements by inheritance); so conversely, a realizational Element embodies each conceptual Element that it realizes. The distinction between them lies in the information exposed, which in conceptual Views is topic-specific, and in realizational Views is item-specific.

Just as one might start with something real and treat it from different conceptual points of view, one might also bring different conceptual points of view into convergence in some real thing. In reality, both approaches evolve in partnership, with no a priori precedence between conceptual and realizational Views (as for instance in concepts always being developed before realizations, which is practically impossible). From an architectural point of view, conceptual and realizational Views are concurrently progressing peers, each needing to be reconciled routinely with the other. This reconciliation is inevitably iterative, as developments in each prompt evolution in others. Co-development of conceptual and realizational Views often progresses by expanding each other's scope, especially during early development. Realization may expand to meet newly identified conceptual needs. Conceptual scope must broaden as new realizations are added.

As peers, the concept–realization mapping is navigable in both directions. That is, a realizational View must be cognizant of all conceptual Elements that are realized by any subject realizational Element; and conversely, a conceptual View must be cognizant of all realizational Elements that realize its conceptual Elements. This mapping is consequently a central means (the other being by top-down conceptual elaboration) by which lateral architectural connections are made and mutual consistency is established. In this essential binding role, realizational Views bring together what separation of concerns among conceptual Views has kept apart, and they ground in actionable Requirements the constraints that have converged in deliverable products.

An essential feature of a good architecture is tolerance to variation, whether such variation arises from acquisition choices, development uncertainties, advancing scope, manufacturing or performance tolerances, measures taken for decoupling, margin, or other reasons. Therefore,

when a realizational View is about a product, it is generally not about a specific design or implementation. Rather, the intent is to reflect and embrace the range of possibilities (different designs, parameters, configurations, providers, or whatever) that are declared acceptable within the conceptual architecture. A realizational View narrows what the conceptual Views allow *only* to the extent reflected in conceptual mapping choices (i.e., which concepts map to which realizations). Should that seem inadvisably generous, there is likely a neglected gap in conceptual considerations that would have supplied the neglected constraints.

The same principle applies when a realizational View is about a specific, preexisting, *external* item, which may nonetheless be subject to uncertainty or change. Whether this is another system or an environmental item, it is still necessary to characterize the range of possibilities that must be accommodated or tolerated.

Following are descriptions for particular realizational Viewpoints. As for conceptual Viewpoints, general guidance above applies, but particular Viewpoints would be intended to provide more directed advice.

There are likely to be more realizational Views than conceptual Views, given that conceptual breakdowns overlap in a variety of ways across a realizing system.

Collections of realizational Views are typically organized by hierarchical product breakdown, by staged deployments, by operational activities, and so on, depending on the nature of the items addressed by the realizational Views.

3.2.1 Composition: The following Viewpoints address the hierarchical division of a system into its separate implementation items and their primary deployed compositions for the purpose of creating assignable units of responsibility.

Product Breakdown

To be provided

Flight System — *To be provided*

Ground System — *To be provided*

Integration and Test System — *To be provided*

Science System — *To be provided*

Modularity — *To be provided*

External Services

To be provided

Telecommunication and Radiometric Services — *To be provided*

Assembly and Test Facilities — *To be provided*

Transportation Services — *To be provided*

Launch Services — *To be provided*

Environment

To be provided

Dynamic — *To be provided*

High-energy Radiation — *To be provided*

Thermal Effects — *To be provided*

Pressure — *To be provided*

Electromagnetics — *To be provided*

Celestial Bodies — *To be provided*

In situ Characterization — *To be provided*

Other hazards — *To be provided*

3.2.2 Science Datasets: Views governed by "science dataset" Viewpoints address the penultimate objective of most space missions, which is the science data they return. These Views address its planning relative to science objectives, its collection and processing for subsequent analysis, its curation and archiving, and its distribution. The quality and completeness of these datasets relative to mission science objectives is a key aspect of science traceability for a project.

3.2.3 Mission Plan: Views governed by "mission plan" Viewpoints describe circumstances during a mission in which the system must perform. The intention established for such Views is not to specify a particular mission plan, but rather to establish a range of possibilities that a given system might encounter, either because particular plans will continue to evolve, or because other uncertainties prevent specificity.

Within the bounds of the mission plan they assert, these Views may also express current *particular* plans, much the way that particular design might be described for a hardware unit. Information of this sort can be useful for point analyses, but always with the understanding that all variations within the bounds of the mission plan must *also* be tolerable. The narrower the bounds drawn by the mission plan, the less tolerant it will be to system variations.

Launch and Trajectory

To be provided

Mission Activities

To be provided

3.2.4 Deployments: Views governed by "deployment" Viewpoints address the different configurations that a system design must support over the course of its development, integration, testing, preparation, and operation.

There can be many such deployments to define. Some deployments will be tailored specifically to testing particular functions or testing in particular simulated environments, where additional support capabilities must also be developed. Some may be deliberately stressful, requiring dedicated qualification units. Some may be narrowed in scope, such as in single string test beds or in software testbeds. Some may require emulators for flight equipment that isn't available or that can't provide realistic behavior under test conditions. Some may exercise interfaces that are used only for test. Some may need particular facility accommodations. Some may add environmental situations that must be tolerated. Some may represent different flight configurations related to articulations, separations, and the like. Some may involve different interoperation configurations in a distributed system. And so on.

Each such deployment involves either a different combination of Elements, or different Relationships among these Elements. Many components besides those deployed eventually in mission operations will be needed. Each deployment (and the components in it) will also be subject to different Requirements, according to its composition and purpose. Therefore, no architecture description is complete without diligent consideration to all deployments that expose significant variations. Documentation of these deployments provide the basis for asserting that the appropriate requirements have been levied on all the various instantiations of system elements.

Integration and Test Configurations

To be provided

Transport and Storage Configurations

To be provided

Operational Configurations

To be provided

■

4 Process

The process of architecting responds to the need to develop a design space for a system that accomplishes a given objective. For JPL missions, the need is typically articulated by the need to collect measurements in support of one or more scientific hypotheses. The approach to collecting the measurements is further constrained by programmatic considerations (cost, schedule), available technology, and the physics of collecting useful data from distant objects. Defining a design space that reconciles and balances these Concerns is the purpose of architecting.

For example, a principal investigator (PI) can make a case to NASA to fund the collection of a certain dataset of interest. The PI is one Stakeholder, who has Concerns about the type and quality of the data, but the NASA funding source is another Stakeholder, who has Concerns about cost and schedule. These Concerns must be turned into the quantitative success criteria for the project. The architecting process considers different options and approaches for getting the data to be reviewed and assessed; this is a necessary aspect to negotiating an appropriate set of success criteria. This illustrative example uses the top most View of a project, but the same pattern applies to each project component, where each delivering team responds to incoming constraints by developing a design space.

The process description below uses the terminology and follows the guidance from ISO/IEC 15288 [10] and the NASA Systems Engineering Handbook [11]. There are three core ideas that shaped this process: (1) **Framing and Focus:** Separation of problem formulation as a separate activity, deserving of the same focus as synthesis and evaluation; (2) **Lifecycle applicability:** Application of architecting across all phases of the JPL project lifecycle, but acknowledging that the nature and role of architecting activities depends on the lifecycle phase; (3) **Iteration:** Definition of an iterative approach to defining a design space, with a need for quantitative criteria for completing each iteration.

Architecting is not just a method that is applicable at the highest levels of system design; the engineering of a product can be made more rigorous by application of architectural principles (e.g., identification of Stakeholders, iterative problem formulation and synthesis cycles). The process stops when the probability of delivering an Element that satisfies the technical (performance and functional) constraints, within the programmatic (cost and schedule) constraints, is deemed acceptable.

A process architecture can be described in an architecture description. The architecting process for defining a process architecture is therefore outside the scope of this methodology. The approach here is simply to assert this process, as applied to the architecting of systems.

The architecting process described here applies to both technical and programmatic aspects of a development. These are bound by overlapping Concerns of performance, cost, safety, and so on that must be traded against one another.

A brief overview of the NASA/JPL project lifecycle is included below, followed by a delineation of the architecting process, and a description of the role of architecting in each phase of the lifecycle. While this process description is intended for use on JPL flight projects, it is generic enough to be applied in developing any design solution.

4.1 Context: the NASA/JPL Project Lifecycle

The NASA/JPL project lifecycle definition organizes the lifecycle of a project into 2 phases: Formulation and Implementation. Within each of these NASA lifecycle phases, one or more Project lifecycle phases are defined (in addition, a pre-formulation project lifecycle phase is also defined). Each project lifecycle phase is delimited by a Key Decision Point (KDP), where the project maturity is assessed, and a determination is made as to whether the project is ready to proceed into the next phase of development (see Figure 1). Within each phase, there are required Life-Cycle Reviews (LCRs) that "provide a periodic assessment of the program's or project's technical and programmatic status and health at key points in the life cycle" (ref 7120.5E). For each LCR (colloquially referred to "gate reviews"), the JPL Flight Project Practices [12] define the set of Gate Products that a JPL flight project must provide, and the required maturity for each of these Gate Products.

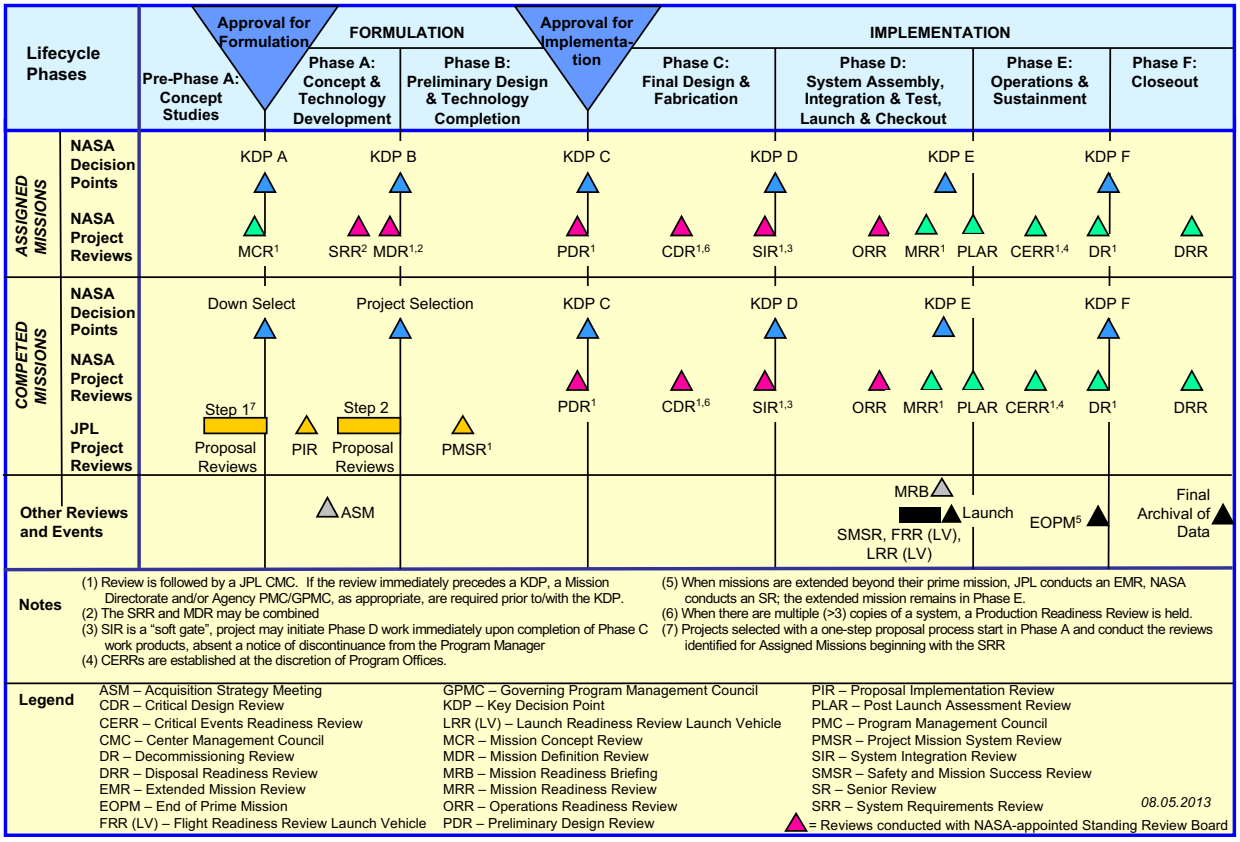


Figure 1: NASA/JPL Project Lifecycle (ref FPPs)

The set and maturity of required Gate Products at each LCR define the end state that the project teams strive for. The iteration cycles described in the following section provide a means for planning the necessary work to result at these end states. Each iteration provides an opportunity to assess the maturity of the project work to date, and an opportunity to adjust the goals and focus of future iterations to ensure the reach the required maturity state. The number of iteration cycles will depend on the needs, size and nature of each project.

The necessary information to generate most Gate Products is captured in the architecture description. While in principle all of the content necessary to produce Gate Products could be

captured in an architectural description, in practice some information and content would remain outside of the architecture description (e.g., high-level plans and agreements, and content which already has purpose-built methods and tools (e.g., network schedules and JPL project-line work agreements)). Where the Gate Product content is derived from the architectural description, the Gate Products are generated by assembling content from the Views and other categories in the architecture framework. Because the content of the architecture description is structured, the content can be retrieved and assembled with standard queries, which can be re-used to produce updates to Gate Products as the content matures. This effectively defines a template for Gate Products that can be reused throughout the project and between projects, resulting in standardized artifacts and reduced effort in artifact generation.

4.2 Core Ideas

The next sections describe a detailed set of steps to apply the principles and abstractions described in earlier section. However, this process description is based on a few core ideas that are easily articulated:

1. **Understand the problem:** Spend the time to interact with all the system Stakeholders to ensure that the system under development is being designed to the appropriate criteria. Document these criteria and revisit these understandings periodically as system development proceeds. Document which Stakeholders care about which constraints (provenance), and why they care (rationale).
2. **Decompose the problem:** Sub-divide the problem into separable aspects that can be addressed individually. This separation of concerns corresponds to the conceptual and realizational Views defined in the previous section.
3. **Bound the solution space:** For each aspect, develop an approach that responds to the constraints. Document the approach in a functional block diagram that captures the relevant Elements, their Relationships with other Elements, and the Function (purpose) of each Element in that context. Document any Trades or Scenarios used in the development of the solution. Document the salient characteristics (Properties) of the Elements and Relationships, and identify/document consequent performance constraints on those Properties (these ultimately become Requirements). Perform analyses to assert that the solution satisfies the initial constraints.
4. **Reconcile solutions:** Assess the consistency of each aspect with all of the other aspects. Perform this assessment by mapping each realizational Element to the appropriate conceptual Elements. Perform trades and adjust the proposed solutions to resolve inconsistencies or implementation issues in realization. Use the results of these assessments to re-negotiate incoming constraints if an adequate solution cannot be found.

These basic ideas are applied in the following process description. While this approach is consistent with the above ideas, should not be interpreted as the only way in which they might be applied.

4.3 Process Description

There are two iteration cycles that organize the steps in the methodology: an outer cycle and an inner cycle. The outer cycle is concerned with problem definition: identifying Stakeholders and Concerns, decomposing the problem, and negotiating updates to Success Criteria. The inner cycle

is concerned with synthesis and evaluation: articulation and assessment of approaches/options, the selection and integration of these options, and the identification of additional constraints that result from this selection. Several iterations of the synthesis and evaluation cycle occur for each iteration of the problem definition cycle.

Organizing this work as a set of iterations is a means to define and schedule the tasks and interim goals needed to achieve the project lifecycle phase end state. By breaking down the effort into a set of iterations, this approach provides points in time during the development process that can be used in assessing the progress of the project team in performance of the work. These assessments can then be used in planning and scheduling work needed in future iterations. Note that not all work can be planned in advance, and in planning goals and objectives for each iteration resources should be reserved for dealing with tasks that are outside of the iteration planning process (e.g., *ad hoc* tasks and Stakeholders/external teams working to different schedules).

The result of the architecting work is an architecture description of the system. The process of successive iterations is an additive process that completes when the probability of delivering an Element that satisfies the technical (performance and functional) constraints, within the programmatic (cost and schedule) constraints, is deemed acceptable. At this juncture, the definition can be handed off to a developer/supplier to provide. After this point, as the design, implementation, and operations processes proceed, it is beneficial to assess the adequacy of the defined architecture (e.g., Were the margins, in fact, adequate? Were all of the concerns adequately addressed by the end product?), and document these findings so that future projects can leverage this information. Organizations responsible for viewpoints may choose to update the definitions, or adjust the related processes to improve future application.

4.3.1 Problem Definition Cycle

The focus of this activity is to ensure that the system objectives are clear, the initial constraints are known and the Concerns of all of the Stakeholders are adequately addressed. At the beginning of a project the object may be stated in an unclear fashion, and only a subset of constraints may be explicitly declared. In addition, it may be unclear if a feasible solution exists to satisfy the stated need. The problem definition cycle provides a structured way to engage the Stakeholders and provide direction to the project team. The project leadership decides on the number of problem definition cycles to perform within each project lifecycle phase.

Gate reviews are part of the NASA project lifecycle. Generally, one or more problem definition cycles are performed between gate reviews. Gate reviews have particular criteria which must be satisfied to move to the next project phase, characterized by the delivery of documents that have achieved specific levels of maturity. These criteria determine the set and maturity of the Views that must be completed by the gate review. Content in the architecture description evolves and matures, but “concept” (or “formulation” or “preliminary design”) is not a measurable stage of development. We need definitive criteria for gauging progress. Reviews play a vital role where objectivity is problematic.

Steps in the problem definition cycle

1. Identify Stakeholders and engage them to refine the nature of the problem and determine significant aspects of the mission.
    a. Example Stakeholders are the project scientist/PI and the funding source (NASA/SMD).
    b. The significant aspects that each Stakeholder is interested in are documented in Concerns. These are the motivating issues for the project.
    c. Performed by: Project Manager (PM), Project Scientist (PS) and Project System Engineer (PSE)
    d. Needed Input: initial statement of need
2. Review and agree on current set of mission objectives and constraints (e.g., mission success criteria, L1 requirements).
    a. These are the mission Success Criteria.
    b. Initially, the set is based on the judgement of the involved engineers and managers, but is informed and shaped by work performed in subsequent iterations.
    c. Performed by: Project Manager (PM), Project Scientist (PS) and Project System Engineer (PSE)
    d. Needed Input: List of Stakeholders and their Concerns
3. Identify the set of conceptual and realizational Views needed to capture the response to the current set of Success Criteria, and to assess feasibility at this stage in the project lifecycle.
    a. While each project is unique, many of the Stakeholders and Concerns will be the same from project to project. Therefore, Viewpoints that respond to these Concerns can be re-used. New Viewpoints can be defined if existing Viewpoints do not cover the set of Stakeholder Concerns.
    b. Typically, the set of Views is drawn from the established norms for dividing the work with good separation of concerns—Mission Design, Systems Engineering, Project Management, etc.—that are part of the JPL guidance for flight projects (institutional WBS, Flight Project Practices, etc.).
    c. The set of realizational Views will increase as the architecture description becomes more detailed (as additional products are defined at lower levels of design).
    d. Assign View development responsibility to an individual or team.
    e. This is a subset of the Views needed for the system as a whole.
    f. The content and maturity of these Views depends on the point in the project lifecycle and focuses on key questions that need to be resolved.
    g. Performed by: Project System Engineer (PSE)
    h. Needed Input: Project Success Criteria, pre-defined Viewpoints
4. Assess the progress of any prior Problem Formulation cycles, and determine the goals of this iteration, given the required state at the next LCR.
    a. These objectives provide direction for the set of synthesis cycles.
    b. Performed by: Project System Engineer (PSE)
5. Perform one or more iterations of the synthesis cycle, working in the Views and auxiliary views (Models, Analyses) to communicate and capture descriptions of project Elements and their constraints.
6. Use the outcome of this work to assess compliance with the set of mission Success Criteria.
    a. Compliance is shown through the results of Analyses.

   b. The documented approach to meeting the Success Criteria, and resulting analysis, provides a basis for negotiating updates to Success Criteria. Early in the mission lifecycle changes to Success Criteria typically come about due to a better understanding of the problem or solution space, or from changes to the statements of need. In later phases, it is sometimes necessary to update Success Criteria due to problems in the Implementation phases, including failures during operations.
   c. Performed by: Project System Engineer (PSE)
7. Repeat cycle until the Success Criteria reach the required maturity level for the current stage of the project.
   a. e.g., at MCR, need PRELIM L1 requirements.
   b. Should changes to the set of Success Criteria occur in the Implementation phases, new iterations of the problem formulation cycle may need to be planned in response.

4.3.2 Synthesis Cycle

The goal of this activity is to ensure that the definition of project Elements is captured in a consistent and structured way. As different approaches to meeting the mission success criteria are introduced and assessed, they are captured in the appropriate Views. As the set of Trades are identified and completed, the resulting Elements and their necessary characteristics (including constraints), are documented. Each iteration of the synthesis cycle focuses on a particular subset of the work to go, allowing both a mechanism for planning this work and a way to react to changes or incomplete tasks from prior iterations. The goal is to reach a satisfactory solution (design space) to the current set of defined project success criteria appropriate for a particular point in the project lifecycle. Iterations of the synthesis cycle early in the project lifecycle inform the selection and set of success criteria, whereas later iterations serve to apply the constraints to the full set of technical and programmatic domains.

Steps in synthesis cycle:

1. Define goals of the iteration cycle (completion state of Views and other products).
   a. In support of problem formulation goals and required artifact maturity at next LCR
   b. Review current completion state of each conceptual View.
      i. Open Trades, incomplete information
   c. Review current completion state of each realizational View
      i. Open Trades, incomplete information
   d. Assess progress to date and progress needed to achieve maturity by next LCR.
      i. Adjust goals of current and future iterations accordingly
2. Determine the set of identified conceptual and realizational Views to be addressed in this iteration.
   a. The set and expected maturity of each View will depend on how the project has planned to work of the project teams.
   b. Map objectives and constraints to each View.
   c. The Conceptual View structure is expected to remain static from project to project, whereas the set of realizational Views will vary.

3. Within each conceptual View, define a set of alternatives (possible approaches) to satisfying the proposed constraints.
    a. Articulate each alternative in a functional block diagram (FBR).
        i. Use the FBDs to identify conceptual Elements, Relationships, and Functions.
    b. Document any Scenarios needed to describe intended Element interaction.
    c. Identify characteristics of the Elements (Properties and constraints) needed to provide the required function/performance.
    d. Trade the alternatives, and use to make decisions on approach, or negotiate on the incoming set of constraints.
        i. A Trade is assessing candidate sets of new constraints.
4. For each realizational view, define a set of real alternatives that utilize the alternatives specified in the conceptual views.
    a. For each alternative, identify the conceptual options that are used in the definition of that alternative. This may be captured in a realizational Trade.
    b. Within each alternative, the Function of each defined real Element must be identified. This is done by making choices about how the set of conceptual elements are realized, and may also may be captured in realizational Trades. These Trades may result in reconsideration of the associated conceptual approaches. It is expected that there will be an interplay between the realizational alternatives being considered in this step, and the conceptual alternatives being considered in the prior step.
    c. Document the Elements, interfaces (or other Relationships), Functions, Scenarios and constraints on the Elements in each alternative.
    d. Assess the alternatives for implementation feasibility. This may trigger additional realizational Trades, or requests for new conceptual alternatives to be developed.
        i. implementation feasibility = we can actually build a system to perform these Functions within the cost/schedule constraints
5. Evaluate the option space, and select a specific option in each conceptual View, and a specific realization.
    a. This establishes a new baseline, that includes a set of derived constraints that must be satisfied to ensure satisfy incoming constraints.
    b. Assess completeness of selected approaches, document work to go (this can be automated with pattern completeness checks).
    c. Identify drivers and issues with incoming constraints.
    d. Generate artifacts to capture current baseline, as derived from selected options in conceptual and realizational Views.
6. Assess accomplishments in cycle vs goals.
    a. Identify incomplete work, and determine where/whether to include in future iteration cycles.
7. Repeat cycle as needed to reach a satisfactory solution (design space) to the incoming constraints.

4.4 Application Across the Mission Lifecycle

The sections below describe the role of architecting in each project lifecycle phase, the set of LCRs in that phase, and the set of architectural Views needed to support the systems engineering gate products required at each LCR.

Content in the following sections are taken from the following sources:

- Descriptions of mission lifecycle phases are taken from the NASA Space Flight Program and Project Management Handbook [13]
- Descriptions of lifecycle reviews ("gate reviews") are taken from the NASA FY2021 Budget Request Executive Summary [14] and from the JPL Institutional Project Review Plan [15]

Each section below includes a summary of the architecting process and the key categories of the architectural description needed at each gate review. For each gate review, a detailed listing that correlates all the gate products to the architectural Views that contribute content to these products is in Appendix C.

4.4.1 The Role of Architecting in pre-Formulation

Pre-Formulation activities involve Design Reference Mission analysis, feasibility studies, technology needs analyses, engineering systems assessments, and analyses of alternatives that typically are performed before a specific project concept emerges. Pre-Formulation activities include identifying risks that are likely to drive the project's cost and schedule and developing mitigation plans for those risks. Pre-Formulation activities include only a single project lifecycle phase: Concept Studies.

The pre-formulation phase is the most important phase for architecting. This is the phase where decisions are made that shape the direction and success of the entire project, and the decision process must be done in a principled and transparent manner. The primary focus in this phase is problem formulation (understanding the purpose of the mission and associated constraints via interaction with stakeholders) and defining the largest possible design space. While a point design is often developed in this phase, its purpose is to assert evidence of a feasible solution, rather than providing a basis for design iteration. Architecting provides a means to connect stakeholder concerns to the definition of the design space. Documenting these connections, *and* their rationale, must be part of the products generated in the architecting process. Analyses are used to produce evidence that the characteristics of the selected design space are sufficient to meet the negotiated mission success criteria.

Concept Studies Phase Life-cycle Reviews

Mission Concept Review — The Mission Concept Review (MCR) evaluates the feasibility and merit of the proposed concept(s). Further, it examines the maturity of the concept(s) and associated planning to determine if the definition is adequate to begin formulation.

To support the objectives of the MCR, the architecture description captures much of the content needed for MCR gate products (see Table 1). A few key examples are provided below:

- Preliminary Level 1 requirements, with supporting rationale and analysis, in the Stakeholders and Concerns categories
- A self-consistent set of Preliminary Level 2 project Elements, their Functions, interfaces and operating Scenarios in the Project System realizational View
- Project work breakdown structure (WBS), and the assignment of defined project Elements to the WBS in the Work Breakdown conceptual View

- Preliminary Level 2 (Project, Science) and Level 3 (Spacecraft, Payload) requirements, with supporting rationale and analysis, traceable to and consistent with Level 1 requirements, queried from the element realizational Views

See Appendix C for the full set of architectural Views and content in the Architectural Description, and the mapping to the required gate products.

4.4.2 The Role of Architecting in Formulation Phases

Project Formulation consists of two sequential phases, denoted as Phase A (Concept and Technology Development) and Phase B (Preliminary Design and Technology Completion). During Formulation, the project explores the full range of implementation options, defines an affordable project concept to meet requirements, and develops needed technologies. The activities in these phases include developing the system architecture; completing mission and preliminary system designs; acquisition planning; conducting safety, technical, cost, and schedule risk trades; developing time-phased cost and schedule estimates and documenting the basis of these estimates; and preparing the Project Plan for Implementation.

The Formulation phase is the phase most associated with system architecting. Not only are the top-level elements and interfaces baselined, but preliminary design responses to the architecture are developed and assessed. Many of the products typically associated with "architecting" are produced in this phase: mission and system element descriptions, requirements documents at Levels 1-4, interface requirements documents, compliance assessments (design report) and risk assessment (e.g., PRA, FT, FMECA). The methodology described in this document ensures that all the relevant aspects of a mission are included in the formulation of the architecture, and that these aspects are properly reconciled in a realizable system. In particular, the development and documentation of Trades is critical, as it is this assessment of alternatives that is the bulk of the engineering work occurring in this phase.

An architecture description provides a living repository in which engineering teams may perform their work, and it is the source of content when a document release is needed. The methodology supports iterative spiral development: successive stages of constraint definition and refinement that are assessed at each review. The use of Viewpoints provides a time-saving element in document generation, as they provide a consistent basis for templates for gate products and other deliverables. If both the architecture and the design response are captured in an accessible form, then automated generation of resource margin reports, risk assessments, requirements compliance (design report) and margin assessments can be conducted.

By the end of the phase, sufficient analyses have been performed to assert acceptable levels of residual risk (e.g., acceptable approach, margins and reserves) to proceed with the implementation phase.

Phase A Life-Cycle Reviews

The purpose of Phase A is to develop a proposed mission/system architecture that is credible and responsive to program requirements and constraints on the project, including resources. The Phase A work products need to demonstrate that the maturity of the project's mission/system definition and associated plans are sufficient to begin Phase B, and the mission can probably be achieved within available resources with acceptable risk.

System Requirements Review — The System Requirements Review is the lifecycle review in which the decision authority evaluates whether the functional and performance requirements defined for the system are responsive to the program's requirements on the project and represent achievable capabilities. [14]

The System Requirements Review evaluates the project requirements to determine if they are responsive to the program's requirements on the project and represent achievable capabilities. This review also evaluates the status of the requirement decomposition, and flow-down, and the plans for completing the requirements definition. [15]

To support the objectives of the SRR, the architecture description captures much of the content needed for SRR gate products (see Table 2). A few key examples are provided below:

- Baseline Level 1 requirements, with supporting rationale and analysis, in the Stakeholders and Concerns elements. These capture the discussions, particularly between the sponsor (e.g., NASA Planetary Science Division) and the project on the primary project constraints.
- A self-consistent set of Baseline Level 2 and Level 3 project Elements, their Functions, interfaces and operating Scenarios in the Project System realizational View
- Baseline Level 2 (Project, Science) and Level 3 (Spacecraft, Payload) requirements, with supporting rationale and analysis, traceable to and consistent with Level 1 requirements, queried from the element realizational Views
- Key/Driving Level 4 requirements queried from Level 4 element realizational Views
- Baseline Systems Engineering approaches and policies, documented in the Systems Engineering conceptual View

Mission Definition Review (leads to KDP-B) — The Mission Definition Review is the lifecycle review in which the decision authority evaluates the credibility and responsiveness of the proposed mission/system architecture to the program requirements and constraints on the project, including available resources, and determines whether the maturity of the project's mission/system definition and associated plans are sufficient to begin the next phase, Phase B.

KDP-B is the lifecycle gate at which the decision authority determines the readiness of a program or project to transition from Phase A to Phase B. Phase B is the second phase of Formulation and means that:

- The proposed mission/system architecture is credible and responsive to program requirements and constraints, including resources;
- The maturity of the project's mission/system definition and associated plans is sufficient to begin Phase B; and
- The mission can likely be achieved within available resources with acceptable risk.

The Mission (Instrument) Definition Review evaluates the preliminary planning, requirements, mission and system (instrument) descriptions, and estimated life-cycle cost to assess the maturity of the project and the progress made in defining the mission (instrument). Additionally, the MDR (IDR) evaluates the project's plans to determine if they are sufficient to transition to Phase B. (ref. Institutional Project Review Plan, Rev. 9, JPL Rules! DocID 75512).

To support the objectives of the MDR, the architecture description captures much of the content needed for MDR gate products (see Table 3). A few key examples are provided below:

- Preliminary System Safety and Mission Assurance Plans derived from the Mission Assurance conceptual View
- Preliminary Mission Operation Concept document, which uses as its basis the integrated Scenarios from the Project System realizational View
- Preliminary Environmental Requirements Document derived from environmental constraints and assumptions captured in the Environmental Compatibility conceptual View
- Preliminary Design Report, captured as a design compliance Analysis, that utilizes the information in the architecture description (structure, composition, requirements) to assess the preliminary design descriptions
- Preliminary Launch Services Requirements document, consistent with the needs, elements and interfaces of the mission
- A Phase B Project Task Plan drawn from Management Structure and Work Breakdown Views that are consistent with the project elements, and institutional guidance

Phase B Life-Cycle Reviews

The purpose of Phase B is for the project team to complete their technology development; engineering prototyping; heritage hardware and software assessments using the Systems Engineering Handbook, NASA/SP-2007-6105 Rev 1, Appendix G; and other risk-mitigation activities identified in the project Formulation Agreement and to complete the preliminary design. The project demonstrates that its planning, technical, cost, and schedule baselines developed during Formulation are complete and consistent; the preliminary design complies with its requirements; the project is sufficiently mature to begin Phase C; and the cost and schedule are adequate to enable mission success with acceptable risk.

Preliminary Design Review (leads to KDP-C) — The Preliminary Design Review is the lifecycle review in which the decision authority evaluates the completeness/consistency of the planning, technical, cost, and schedule baselines developed during Formulation. This review also assesses compliance of the preliminary design with applicable requirements and determines if the project is sufficiently mature to begin Phase C.

KDP-C is the lifecycle gate at which the decision authority determines the readiness of a program or project to begin the first stage of development and transition to Phase C and authorizes the Implementation of the project. Phase C is first stage of development and means that:

- The project's planning, technical, cost, and schedule baselines developed during Formulation are complete and consistent;
- The preliminary design complies with mission requirements;
- The project is sufficiently mature to begin Phase C; and
- The cost and schedule are adequate to enable mission success with acceptable risk.

The project Preliminary Design Review evaluates the project's maturity and readiness to proceed with implementation. This review evaluates the completeness and consistency of the planning, technical, and cost baselines developed during formulation. It assesses the compliance of the design with applicable requirements.

To support the objectives of the PDR, the architecture description captures much of the content needed for PDR gate products (see Table 4). A few key examples are provided below:

- Baseline Project Verification and Validation Plan, with traceable compliance to the SEMP and relevant institutional guidance, and documenting types of equipment and processes, in the Project & System Level Functional V&V conceptual View
- Preliminary Science Data Management Plan, drawn from the Science Data Management conceptual View, and consistent with the defined project science datasets, defined systems and operating Scenarios
- Baseline office implementation plans, with traceable compliance to the project policies, the SEMP, and institutional guidance via office-specific development conceptual Views
- Baseline compliance matrices for the JPL Flight Project Practices, Design Principles and Systems Engineering Practices, with traceable compliance from the policy statements to project technical elements and programmatic processes
- Baseline Spacecraft Flight Software requirements document queried from the Flight Software realizational View
- Baseline Flight System I&T Plan, describing the needed equipment, deployments and overall I&T flow, as Elements and Scenarios in the FS I&T realizational view, and consistent with the project elements and V&V policies

4.4.3 The Role of Architecting in Implementation Phases

Project Implementation consists of Phases C, D, E, and F. During Phase C (Final Design and Fabrication) and Phase D (System Assembly, Integration and Test, Launch and Checkout), the primary activities are developmental in nature, including acquisition contract execution. Phase C includes completion of final system design and the fabrication, assembly, and test of components, assemblies, and subsystems. Phase D includes system assembly, integration, and test; prelaunch activities; launch; and on-orbit checkout (robotic projects) or initial operations (human space flight projects). All activities are executed according to the Project Plan developed during Formulation. KDP E marks approval to launch. After successful on-orbit checkout or initial operations, the project transitions to Phase E. The start of Phase E (Operations and Sustainment) marks the transition from system development and acquisition activities to primarily systems operations and sustainment activities. In Phase F (Closeout), project space flight and associated ground systems are taken out of service and safely disposed of, although scientific and other analyses might continue under project funding.

In Implementation the role of system architects and architectural description shifts from the active generation and assessment of the system architecture to assessment of selected design choices and reactions to unanticipated circumstances. In the latter case, a well-documented system architecture provides the necessary context and tools to assess options and implications when design choices or implementation plans must change or have unexpected consequences. Since these changes in direction usually occur at times of significant daily expenditure by a project, there is pressure to quickly arrive at alternative solutions. Without a readily-available repository and expertise to assess the implications of such decisions, there is a greater risk of making a poor choice. In addition, the architectural description provides a resource—in its Views (rationale), Trades, Analyses, etc.—that provides context needed in the development of Implementation phase products such as V&V/I&T plans and procedures. The availability and use of an

architectural description in this phase preserve continuity and consistency of the approach defined in the Formulation phase.

Phase C Life-Cycle Reviews

Project Implementation begins with Phase C as the project team implements the project in accordance with the Project Plan. The purpose of Phase C is to:

- Complete and document the final design that meets the detailed requirements;
- Ensure that the systems engineering activities are performed to determine if the design is mature enough to proceed with full-scale implementation within the constraints of the Management Agreement;
- Perform qualification testing;
- Develop product specifications and begin fabrication of test and flight architecture (e.g., flight article components, assemblies, subsystems, and associated software);
- Develop detailed integration plans and procedures; and
- Ensure that all integration facilities and personnel are ready and available.

Critical Design Review — The Critical Design Review is the lifecycle review in which the decision authority evaluates the integrity of the project design and its ability to meet mission requirements with appropriate margins and acceptable risk within defined project constraints, including available resources. This review also determines whether the design is appropriately mature to continue with the final design and fabrication phase.

The Critical Design Review evaluates the integrity of the project design by reviewing the consistency among the mission, systems, and operations designs that incorporate the results of lower-level detailed designs and test results of early hardware models, plus modeling, simulation, and analysis of the systems' behavior and performance. The capabilities and design margins of the project systems, together with the available programmatic reserves and unused technical resources, are reviewed to assess the risk to perform within sponsor-imposed constraints. The status of the project's significant risks, the effectiveness to date of the safety and mission assurance programs, and the progress made against management plans are also reviewed.

To support the objectives of the CDR, the architecture description captures much of the content needed for CDR gate products (see Table 5). A few key examples are provided below:

- Baseline Design Report, captured as a design compliance Analysis, that utilizes the information in the architecture description (structure, composition, requirements) to assess the detailed design descriptions
- A Phase CD Project Task Plan drawn from Management Structure and Work Breakdown Views that are consistent with the project elements and institutional guidance
- Baseline Mission Plan drawn from the integrated Scenarios in the Project System and other element realizational Views
- Baseline Navigation Plan, with traceable compliance to relevant requirements, policies and mission Scenarios, as captured in the Navigation conceptual View and consistent with elements defined in the Ground System realizational View

System Integration Review (leads to KDP-D) — The System Integration Review (SIR) is the lifecycle review in which the decision authority evaluates the readiness of the project and associated supporting infrastructure to begin system assembly, integration, and test. This lifecycle

review also evaluates whether the remaining project development can be completed within available resources, and determine if the project is sufficiently mature to begin Phase D.

KDP-D is the lifecycle gate at which the decision authority determines the readiness of a project to continue in Implementation and transition from Phase C to Phase D. Phase D is a second phase in Implementation; the project continues in development and means that:

- The project is still on plan;
- The risk is commensurate with the project's payload classification; and
- The project is ready for assembly, integration and test with acceptable risk within its Agency baseline commitment.

The System Integration Review (SIR) evaluates the project's plans for the remaining pre-launch development including whether available resources are adequate and the project's readiness to start system assembly, I&T, and the planning for launch site operations (instrument system integration, test, and calibration). The SIR also reviews the requirements and plans for and status of the project verification and validation (V&V) activity

To support the objectives of the SIR, the architecture description captures some content, less than prior reviews, since much of the content needed for SIR gate products (see Table 6) is outside of the architecture description. Examples of content that is included in the AD are:

- Updates to the Science Data Management Plan
- Updates to the Design report
- Updates to the MOS Functional design document
- Updates to the Flight System I&T Plan

Pre-Ship Review — The Pre-Ship Review (PSR) evaluates the completeness of the flight system (or instrument) test program, and assesses the readiness to proceed with shipment of the flight system to the launch site (or instrument to the spacecraft contractor facility), and I&T at the next higher level of integration. The review also assesses the progress toward completion of the development activities planned before launch.

To support the objectives of the PSR, the primary role of the architecture description is to provide content (system descriptions, trades, analyses) for the Certificate of Flight Readiness. The full set of PSR gate products is contained in Table 7.

Phase D Life-Cycle Reviews

The purpose of Phase D is to perform system AI&T; complete validation testing; finalize operations preparations; complete operational training; resolve failures, anomalies, and issues; certify the system for launch; launch the system; and complete on-orbit system checkout.

Operations Readiness Review — The Operations Readiness Review evaluates the readiness of the MOS (Instrument Operations System (IOS)) and all of its component Elements, e.g., GDS (Science Data System (SDS)), to support launch and flight operations.

To support the objectives of the ORR, the architecture description captures some of the content needed for ORR gate products (see Table 8). A few key examples are provided below:

- A Phase E Project Task Plan drawn from Management Structure and Work Breakdown views that are consistent with the project elements, and institutional guidance

- A Decommissioning/Disposal Plan with traceable compliance to relevant requirements, policies and Scenarios, as captured in the Project System realizational View
- Updates to the Navigation Plan

Mission Readiness Review (leads to KDP-E) — The Mission Readiness Review (MRR) evaluates the readiness of the project and all project systems to support launch and the mission.

In this initial assessment, none of the gate products for the MRR are expected to derive from the Architecture Description. The full set of MRR gate products is contained in Table 9.

Phase E Life-Cycle Reviews

During Phase E, the project implements the Project Plan/Missions Operations Plan developed in previous phases. Mission operations may be periodically punctuated with Critical Event Readiness Reviews (CERR), e.g., a trajectory correction maneuver or orbit insertion maneuver.

Below is a list of Phase E life-cycle reviews, but a description of the role of the AD in these reviews has been deferred to a future version of this document:

- Post-launch Assessment Review (PLAR)
- Critical Event Readiness Review (CERR)
- Decommissioning Review (DR)

Phase F Life-Cycle Reviews

During Phase F, the project implements the Decommissioning/Disposal Plan developed and approved in Phase E. The project dispositions all spacecraft ground systems, data, and returned samples, including safe and adequate disposal of the spacecraft. The project team dispositions other in-space assets. The project team closes out all project activities in accordance with the Decommissioning/Disposal Plan.

Below is a list of Phase F life-cycle reviews, but a description of the role of the AD in these reviews has been deferred to a future version of this document:

- Disposal Readiness Review

4.5 Application at Increasing Levels of Design

Architecting is a method that is generally applicable, not just at the highest levels of system design. The engineering of any product, regardless of the level of detail, can be made more rigorous by application of architectural principles (identification of stakeholders, iterative problem formulation and synthesis cycles, etc.).

As an example, below is a brief JPL example of the interface between o Flight system engineering and subsystem engineering:

- FSE team generates constraints (L4 requirements) on the spacecraft avionics
- Avionics team develops a set of options, based on these constraints and concerns of other identified Stakeholders (e.g., line management, regarding workforce availability)
- Review and assessment of options includes both the FSE team, and the other identified stakeholders

- Discussion and negotiation of requirements occurs
- Upon agreement, the baseline set of requirements and approach are documented, along with any remaining issues/concerns to be addressed in future work

■

5 Training, Tools, etc.

The methodology described here is practicable only when those expected to practice it are prepared and properly supported in its application. Therefore training, tools, and other support to practitioners is a key aspect of this methodology.

5.1 Core Ideas

To understand any methodology, it is necessary, not just to learn it as a pedagogical exercise, but to practice it in realistic circumstances on practical applications with expert guidance and critique to keep everything on track. Similarly, to the extent that tools are needed to support a methodology, it is necessary, not just to learn how to use the tools, but to appreciate concepts that they manifest and techniques for exploiting them effectively. And finally, in both respects, it is necessary, not just to address the deployment of this methodology, but also to establish expectations for the quality, coverage, and efficiency of its results.

These considerations prompt a rather involved approach to training for practitioners in this methodology. Classes are the starting point for this training, but they serve mainly as an introduction to key concepts and their motivations. These must be accompanied, initially by dialog to clarify issues and explore their implications, but then directly by hands-on experience with realistic applications, all supported by rich documentation that explains in depth the ideas involved.

In this process, both simple and complex worked examples are needed to facilitate understanding of the products to be developed, the steps needed to develop them, and the metrics used to assess their status. Practitioners must appreciate more than just the desired end state, so training necessarily includes examples that reflect various stages in a project's evolution, with exercises to explore issues of moving from one level of maturity to another.

Experience, capabilities, habits, and required contributions will differ markedly among those asked to learn this methodology. The combination of lessons, examples, documentation, and coaching addresses this diversity to some extent. Nonetheless, there will always be gaps and shortfalls, where training can't provide all of the needed support. Therefore, the potential need for ad hoc support is must be considered in deploying this methodology, especially in its initial utilizations.

Finally, as proficiency is gained with this methodology, it will be necessary to adapt to lessons learned and to new ideas for improving its effectiveness or reach. Collection of feedback and incorporation of these observations into future deployments, including training and tools, is essential to the performance of this methodology.

5.2 Training Requirements

First and foremost, it is important to state that the objective of training is to create a shared understanding of principles, concepts, properties, and relationships. This shared understanding is essential to ensure that collaboration is effective and efficient across disciplines within a

project and across projects over time. While training may incorporate elements of instruction in the mechanics of using specific tools and services, these are incidental details that have no meaning or value except in the context of process objectives.

By way of illustration, consider the result of effective training in digital filter design. We would expect a person who completed such training to demonstrate competent understanding of a variety of topics, including the definition of a filter, the basics of difference equations, causality, impulse response, the difference between infinite and finite impulse response, magnitude response, phase response, etc. These concepts can be explained (and learned) using nothing more than paper and pencil. Application in practice typically involves specialized software tools that solve synthesis problems that are beyond the practical ability of humans. But facility with those tools is not a substitute for deep understanding of the principles involved.

Systems engineering is no different–except for the facts that systems is relatively broad and relatively new so that its mathematical underpinnings are not as strong. Systems engineering can, however, be practiced with a great deal more formal rigor than is common at JPL at present; the methodology described in this document is one step in a long process of raising the bar of rigor.

With that in mind we can lay out some requirements of an effective architecture training program.

The objective of architecting training should be to equip engineers to do architecting, not to become experts in architecting methodology. The distinction is fundamentally about what engineers do because they understand it to be good practice vs. how they conceive it to fit into an abstract formal framework.

Consider pilot training. Trainees are given an introduction to flight dynamics because it explains how certain phenomena are coupled (air temperature and lift, for example) in a way that informs strategy for maintaining control of the aircraft. They are given taught how to use both visual examination of the flight environment (terrain, horizon, weather, traffic) and instruments to make observations of speed, altitude, attitude, rates, etc.

A well-trained pilot integrates all this information in such a way that she can confidently execute a control strategy for landing under ideal conditions as well as in a crosswind with traffic and natural obstacles.

Much of the theory of architecting applies here:

- There are Stakeholders: pilot, crew, passengers, owner, people on the ground, etc.
- Those Stakeholders have Concerns: safety, comfort, travel time, cost, etc.
- There are various Viewpoints associated with those Concerns: airspeed, altitude, attitude, separation, efficiency, etc.
- There are many Views: the literal view out the windshield, instrument displays, etc.
- And so on.

A well-trained pilot has an intuitive understanding of these things and their relations. She knows that landing short of the runway is to be avoided. She knows that airspeed affects lift and turn radius. She knows that a turn coordinator provides a view of roll and yaw and their coupling that helps her apply the correct rudder and aileron inputs to execute a standard rate (3°/s) turn.

She knows all these things without ever using the words *Stakeholder*, *Concern*, *Viewpoint*, *View*, etc. Ironically, forcing her to ponder the definition of *Viewpoint* would serve more as a distraction than an enhancement of her knowledge.

Similar considerations apply to mission architecting at JPL. Years of success (and the occasional failure) have led to a set of more-or-less canonical practices that, due to lack of a formal framework, are often executed with less than desirable rigor. Unifying the practices in a formal framework is important for the Laboratory but the individual practitioner need not understand that framework in detail nor how the elements of practice are classified by it. Instead, the individual practitioner benefits from numerous secondary effects of formalism, including

- careful use of controlled vocabulary for concepts, e.g., *component*, *interface*, *function*, *requirement*, etc.,
- careful use of controlled vocabulary for quantities, e.g., *mass*, *temperature*, *spatial resolution*, etc.,
- careful use of controlled vocabulary for relations, e.g., *contains*, *performs*, *presents*, *authorizes*, *supplies*, etc.,
- consistent reuse of description and analysis perspectives, e.g., *Viewpoints*, and
- consistent reuse of description and analysis products, e.g., *Views*.

These desirable attributes of good practice must first be encoded in written line doctrine. That is, the line discipline experts must commit to writing, with examples, instructions that embody them and require compliance of their employees. If something is good practice, we should require it. If not, we shouldn't. Training must follow this doctrine and refer to it often. But the message of the training must be primarily to inform and reinforce principles and practices, not methodology.

Like our well-trained pilot, our well-trained architect is intimately familiar with architecture practices in the domain of space flight, and expects to see (and to produce) a large set of familiar canonical products (i.e., Views) regarding science objectives, scientific measurements, acquisition, information systems, mission design, navigation, attitude control, configuration management, etc. He has an intuitive sense that "this is a good way to do things", reinforced by the fact that the practice is fundamentally regular and rational and largely uniform from one project to another.

To summarize, the doctrine and associated training material must be informed and disciplined by the formal methodology of architecting, but the elements of the formalism need not appear in the training itself. Instead the training should emphasize the principles of space mission architecting and prescribe practices that respect those principles and produce useful products.

The members of a large project team bring a diverse set of skills, experiences, dispositions, and assignments. Training in architecting should consequently be tailored to that diversity so that people know what they need to do to do their jobs well and can easily add to their skills and knowledge as required.

Developing architecting expertise may extend over years, so it is important that training materials be of such granularity that a budding architect may discover a gap in her knowledge, browse available materials to find applicable content, read a few pages of exposition with illustrative examples, perhaps watch a short video, and then return better prepared to her tasks.

Conventional lecture-style training has utility, but just as in pilot training, there are limits to what you can learn and internalize without actually doing the job. Mentor/protégé relationships are key here. Aspiring architects should acquire the basics through course and reference materials, but the deeper understanding comes from observing those who have mastered the skills, supporting those experts by carrying out directed assignments, and acting as lead under the guidance of experts.

It is important to recognize and reward those who have mastered architecting, in part to provide incentives for others to work toward, but also to clarify the responsibility of those with mastery to mentor and consult for others outside their narrow day-to-day jobs responsibilities and to maintain currency of their own skills and knowledge.

Beyond the purely pedestrian knowledge transfer of mentorship, a most important effect is to establish the expectation of excellence and instill in the trainee the idea that we do excellent work, not because some customer or auditor demands it, but because are personally and institutionally committed to it.

5.3 Necessary Tooling Features

In the methodology addressed here, an architecture framework establishes the essential structure of an architecture description, while particular Viewpoints establish key concepts for recurring topic areas. In addition, though, we also need supporting infrastructure (i.e., tools and an environment to host them) to provide the mechanisms by which these ideas can be expressed and related to one another.

Chief among these is formalism of the architecture framework into an ontology, which defines the entities to be described, their properties, and the rules governing their associations with one another. This becomes the basis that enables the systematic and disciplined deployment of automated reasoning, data management, document generation, and other tooling that supports the architecting process. The architecture framework itself is the beginning of such an ontology, but with recognition from the outset that the categories and relations it establishes are only a start. Viewpoints and the architecture descriptions that conform to them extend these ideas to whatever extent is needed to address the topics at hand. These in turn extend to descriptions of detailed design and other matters beyond the scope of architecture. Therefore, tools for architecting must reside amicably within a larger ontology of information.

Viewpoints provide much of this extension, providing recurring guidance that may be used over many projects. Besides general guidance though, they are also an integral aspect of tooling in that their contributions to the architecting ontology become part of the architecting infrastructure. These extensions must be available to users in such a way that the guidance of Viewpoints is apparent in their use, making it easy to follow, while avoiding violations. This applies not just to individual entries into an architecture description, but to the structure overall as well. For instance, common patterns of interaction among the parts of a system should be simple to deploy, whether in primary Views or in the auxiliary views that support them. Furthermore, in order to manage the complexity of an architecture description, there must be straightforward means of controlling scope, hiding lower level information, honoring good separation of concerns, and so on, all of which place demands on the supporting infrastructure.

The content of an architecture description comprises many separate items (Concerns, Views, Scenarios, etc.), but for an architecture description to maintain its integrity over the course of

development, these items must remain consistent with one another. Key notions in this endeavor include adherence to the ontology, an assured single source for any given assertion, timely assessments of constraint violations, and the like, all of which can be automated within a competent implementation. Users should never feel like they need to be careful because their infrastructure is not.

Finally, there must be provisions for controlling the authorship of assertions in an architecture description and for managing their maturation over time. Permissions must be managed at a level that is not onerously detailed, but that is also not so broad as to impede progress while everyone coordinates. Tooling must provide fluid support to processes of delegation, change control, negotiation, and so on.

Broadly speaking, architecting tooling must support four broad categories of activities; we address each briefly.

5.3.1 Authoring

Authoring denotes the creation of novel content, typically in the form of assertions. *Novel* here simply means refers to assertions that were not in effect until asserted by some authority. For example, a decision to employ solar electric power generation for a mission would take the form of a set of assertions about the design and characteristics of the flight system power subsystem that are taken to be true precisely because the cognizant design authority has declared them to be true. All engineering content originates in such declarations.

Because we want authored content, once created, to retain its meaning indefinitely, it is imperative that all such content be traceable to the (sole) authority who declared it and any other necessary provenance (e.g., date, person acting as such authority on that date, etc.) It is also imperative that the assertions themselves employ only the controlled vocabulary of the applicable ontologies (although the provenance may contain natural language narrative). Finally, the authored assertions must be indexed and stored in such a way that every version of every assertion must be locatable by search and retrieved promptly into the indefinite future.

Authored assertions are in a very real sense the primary product of architecting (and systems engineering in general). JPL spends hundreds of millions of dollars annually to make assertions in which our sponsors and the taxpayers can have confidence. We should curate and steward those assertions accordingly.

5.3.2 Integration

Architecting is almost by definition concerned with complex endeavors requiring collaboration among multiple parties in which each party is responsible for authoring some novel assertions, but every party needs to understand some assertions made by some other parties. Integration refers to the collecting and combining of content from multiple sources, reconciliation of possible conflicts, and harmonization of viewpoints. These integrations may correspond to, for example, system design baselines at nominated milestones (e.g., Critical Design Review) but they may just as well correspond to trade options, speculative investigations, change proposals, etc. Integrations may be compared, combined, etc. The tooling must not be limited to *a single instance* of the system design, but instead must permit arbitrary snapshots of various perspectives at various stages of maturity, one of which will customarily be designated the current baseline.

Integration capabilities should also include, as necessary, adapters to mediate interaction with external tools and sources, translating at the boundary from external (possibly proprietary) representations into those using the consensus ontologies.

5.3.3 Analysis

We denote by analysis the extraction of inferences that can be drawn from assertions. It encompasses a spectrum from rudimentary syntax checks to semantic reasoning to discipline-specific simulations and computations of figures of merit.

The technical world today abounds in high-quality mathematical and engineering analysis tools from the very general to the extremely specific. An architectural analysis facility should not attempt to reproduce such capabilities but instead should focus on model transformation capabilities for translating authored architectural descriptions into appropriate analytical problem specifications that can be solved by existing tools.

Analysis results, along with their provenance, should be managed just as authored assertions are: annotated, indexed, and saved indefinitely.

5.3.4 Reporting

Reporting similarly encompasses a broad range of capabilities, from generating a single spreadsheet to producing a professionally-design and typeset technical document with included figures, tables, appendices, indices, etc. Reporting products will include *ad hoc* artifacts to address a particular topic, products generating specifically for expert review that include comments and annotations for and by reviewers, and final, formal gate products mandated by official process.

As is the case with analysis, there is an abundance of powerful tools for reporting in the technical world today. The objective of architecting tooling is to harness those capabilities for our needs.

5.4 Summary

The training and tool principles and features described above combine for some useful operational properties, including

- the ability to build modeling capability incrementally by drawing on past efforts,
- the ability to advance and evolve modeling tools without invalidating past models,
- effective interaction with specialized tools through the use of common knowledge representation standards,
- the ability to model at varying levels of abstraction and reconcile the consistency of those representations,
- direct traceability between model content and its projections into tailored products for team members and reviewers,
- support for concurrent trades at all levels and stages of development, and
- rapid, frequent global assessment of satisfaction of all committed constraints.

■

Appendices

Appendix A Expanded Viewpoint Examples

A.1 Mass & Inertial Properties Viewpoint

As a basic characteristic of all material objects, mass and associated inertial properties play key roles in many engineering considerations.

> For instance, the mass of flight systems and the apportionment of mass among their parts have implications on launch, trajectory design, structure, pointing control, entry, descent, and landing, etc., so these are tracked as technical resources and subject to management (FPP 6.4) and margin (FPP 5.25) policies.
>
> The distribution of mass in items that are rigid (or nearly so) is characterized by center of mass and moments of inertia, which determine how items rotate in response to forces imposed on them (e.g., engine thrust, motor torque, gravity gradient, and others). These can have broad implications to control authority, vibration modes, structural loads, and so on.
>
> The masses of external Elements are also of interest (e.g., celestial body masses that affect the trajectory).

The Mass & Inertial Properties Viewpoint addresses all such cross-cutting issues of mass management as it relates to spacecraft, both during flight and (as necessary) during ground handling. Masses of ground system Elements are not usually managed, except as they relate to flight system handling.

Although mass management issues may arise directly from Concerns, they are more commonly driven indirectly by launch (e.g., injection C_3), mission design (e.g., ΔV), pointing performance or control authority (e.g., reaction wheel torque or momentum capacity), environmental influences (e.g., aerodynamic drag), or other matters in which realized mass is a significant factor. Such constraints arise mainly in other conceptual Views that address topics where mass is significant. As constraints arise in these Views and are directed ultimately to realizational Elements, they are incorporated into Views governed by the Mass & Inertial Properties Viewpoint.

The purpose of Views governed by this Viewpoint is to ensure…

- that all such issues are identified,
- that the motivating constraints for each are clearly stated and negotiated,
- that all contributors to their satisfaction are enumerated and tracked, and
- that control is exercised throughout development in a manner that mitigates risk.

Allocations

Mass management generally involves at least the combined masses of all material objects in a deployed system. Management of this composite property is typically accomplished by subdividing a collective mass allocation among both identified components of the system and reserves (i.e., set-asides for unanticipated changes). More involved inertial properties (e.g., center of mass) follow a similar pattern, except that additional component properties besides mass (e.g., item locations) are involved in calculating the composite inertial property, in which case these additional properties must be tracked along with mass.

Mass and inertial property allocations are established in Views governed by the Mass & Inertial Properties Viewpoint.

Each component of an allocation should be aligned unambiguously with a delegated development authority (typically via some work breakdown structure). Initial allocations may be coarsely defined, and further sub-allocated as the architecture and authority structure mature. Each sub-allocation is typically governed by the authority at that level. Thus, the ultimate depth of sub-allocation would generally correspond to the depth of work breakdown asserted by the architecture. However, this may need to be tailored for some projects, so any View responding to this Viewpoint should declare the method to be applied for that system. Every allocation is treated as a Requirement.

Components contributing to mass and other inertial properties are tracked via tables, one for each mass-related constraint that must be managed. In each case, table entries address all components covered directly or indirectly by the associated constraint. Sub-allocations are addressed in tables of the same form, but these are routinely combined into a single "rollup" table that itemizes the entire allocation tree for the originating constraint.

For a combined mass constraint (e.g., mass carried to ejection by a launch vehicle), each entry in a mass table should…

- identify a deliverable **component** of the composite with the constraint,
- state the mass **allocation** for that component (typically a not-to-exceed maximum, though probabilistic measures are also possible), and
- provide a current **estimate** for mass upon delivery (or other relevant milestone), as declared in component realizations.

The itemization of deliverable components is commonly shared with a **master equipment list**. However, mass management frequently requires lists specific to particular deployments or situations, so these should not be conflated.

Mass estimates should be in a statistically meaningful form that provides an objective and realistic expression of uncertainty (e.g., no hidden distortions) and that supports an objective assessment of risk at each level (e.g., a maturity assessment). In each case, the **basis for estimate** should be explained. If a project chooses to assert standard uncertainties relative to maturity, these should be declared in the View.

For more involved inertial properties, additional properties must be included in allocation tables, and allocations and estimates must address these properties as well. Where allocations address component properties in combination, care should be taken to ensure that all are within the purview of the assigned authority. If this is not possible (e.g., one property being the mounting location provided by another component), then separate entries should be maintained for the separable contributions.

Components are sometimes added to systems exclusively to deal with inertial property uncertainty. A common example is the addition of ballast to control center of mass location. Other matters that can't be completely resolved a priori may also contribute to mass coupling. Common examples would include propellant for delta-V, and mass for chassis or support structure. In these cases, the mass of the added component may need to vary in relation with the masses of other components. Mass redistribution due to kinematic and dynamic effects must also be considered (e.g., articulated appendages, tethered connections, or propellant migration between

tanks). There may also be situations where non-inertial properties are driven by mass (e.g., tank volume as a function of propellant mass). Wherever such dependencies occur, estimates and allocations must be tailored accordingly.

In a component with propulsive capability, the proportion of mass in expendable propellant—the "wet" mass—is tracked separately from the remaining "dry" mass. In a staged system, combined propellant and "dry" masses must generally be tracked separately for each stage. This typically includes the split of adapter mass between launch vehicle and spacecraft, but kick stages, jettisoned items (e.g., probes or covers), and so on must also be considered. Similar considerations may also apply for systems that are subject to externally induced forces. These can include radiation pressures, gravity gradients, and aerodynamic forces (e.g., aerobraking or atmospheric entry and descent).

It this variety of cases, each tracked configuration comprises a deployment of the system, defined as the set of components that it contains, and in some cases the state of those components or their relationships to one another (e.g., articulation, fill fraction, etc.). Not all deployment will be subject to mass or inertial property constraints, but those that are must be managed.

Management

Historically, estimated mass has tended to grow over the course of development. It is necessary to choose a method for handling this issue.

> One way is to produce estimates that include projections of potential growth relative to current understanding and given relevant precedents. In this approach, component estimates are effectively unbiased (i.e., expected mass at delivery), with reserves held only for additions associated with changes to the composition of the system (e.g., new components, different component requirements, or other unanticipated component design changes).
>
> Another approach is to carry margin for "normal" growth at a composite level. In this approach, component estimates account only for present understanding, while allowance for growth is carried at a higher level, either as additional reserves or as a separate margin allocated for growth.
>
> Further variations of these approaches can carry reserves and margins at just the top level or distributed among two or more levels.

Each View governed by this Viewpoint should include a **mass management plan** that defines an approach appropriate to the project. This approach must define under which authority margins and reserves are to be held, and how allocations to margins and reserves are to be adjusted as a function of development maturity (usually tied to major schedule milestones such as project phase transitions). Quantified guidelines for mass management may be found in JPL Design Principles (6.3.2 System Mass Margins), including the convention to be used in reporting margin as a percentage value.

In addition, mass management plans should include provisions for a programmatic response to allocation violations. These could include policies governing the release of reserves (e.g., via liens), identified descopes or plan changes triggered at defined thresholds, or transfer to a larger risk management process where more sweeping changes are considered.

Supporting the mass management effort should be parallel processes for compiling opportunities for improved margin (either reduced need or reduced uncertainty) and for identifying potential threats (e.g., development problems, or potential requirement changes).

Reporting Artifacts

Conformance relative to mass and inertial property constraints is reported on a regular basis throughout development. Current estimates and allocations are reported, along with the resulting risk relative to allocations, reserves, projections based on threats and opportunities, and so on.

Broad reporting requirements are typically asserted in a Systems Engineering Management Plan, or a Technical Resource Management Plan. In addition, narrower reporting may be required in accordance with constraints arising from particular areas (e.g., a Mission Plan or Navigation Plan).

In each case, reported content consists of…

- allocation and estimation tables, as described above (both summary and detailed accounts), accompanied by…
- itemization of changes from the previous report, as well as overall trends,
- lists of opportunities, threats, liens, etc., and
- identification of allocation violations that should prompt a programmatic response.

Supporting data

To be provided

Definitions

The following definitions are typical for composite mass and inertial property budgets. Rollups can be managed via compositional hierarchies.

Prototype Elements [1]	Properties	extends…
Reference Frame	coordinate system [2]	
Mounted Object	mounting frame(s) [3]	Reference Frame
Material Object	mass [4]	
Solid Material Object	inertia tensor [3,4], center of mass location [3,4]	Material Object, Reference Frame
Liquid Material Object	TBD	Material Object, Reference Frame
Gaseous Material Object	TBD	Material Object, Reference Frame

1 *likely imported from more general Viewpoints that address basic physics or geometry*
~~2 *expressed relative to the Element's coordinate system*~~
3 *defined in terms of physical Element features (typically identifiable points, lines, or planes)*
4 *includes a constraint relating this Property to Properties of its members, if composite*

Relationships [1]	Properties	participants
Attachment	6-DOF displacement [2]	two Mounted Objects

1 likely *imported from more general Viewpoints that address basic physics or geometry*
2 *constrained alignment between mounting or coordinate frames*

Other Properties	owner	constraint on
Estimated mass (1)	realizational View	mass (3)
Allocated mass (2)	mass management View	mass (3)

1 one *for each Element addressed by the realizational View*
2 *one per each relevant Element*
3 *of a designated Material Object*

■

A.2 Work Breakdown Viewpoint

Overview

A well-known reference describes the Work Breakdown Structure (WBS) as follows:

> "The WBS is a product-oriented family tree that leads to the identification of the functions, activities, tasks, subtasks, work packages, and so on, that must be performed for the completion of a given program. It displays and defines the system (or product) to be developed, produced, operated, and supported, and portrays all of the elements of work to be accomplished. The WBS is **not** an organizational chart in terms of the project personnel assignments and responsibilities, but does represent and organization of work packages prepared for the purposes of program planning, budgeting, contracting, and reporting." [16]

Note immediately that the role of the WBS in financial matters (budgeting, reporting) is secondary to its role in planning and organizing the *work* to be performed. (That's why it's called the *Work* Breakdown Structure.) In that sense it is fundamental, not incidental, to the practice of systems engineering. It does, of course, play a role in financial matters, but that is primarily because the most rational way to plan budgets and accrue costs is to align the accounting structure with the actual structural decomposition of the work to be performed. Financial structures follow work structures, not the other way around.

A simple endeavor that can be successfully accomplished by one person or a small cooperating team does not necessarily require a WBS, and it almost certainly does not require a hierarchical WBS with multiple levels. More complex enterprises, however, are characterized by the need to divide the work up in ways that can be delegated to teams, each team having considerable discretion in the execution of its tasks, including the further subdivision of those tasks into subtasks and their assignment to sub-teams. This leads to the familiar tree-structured WBS. We say each node in this tree is an *Authority*, meaning that it represents the power to *author* (that is, to create). The edges in the tree represent delegation of authority; a higher-level Authority assigns some subset of its authoring power to a lower-level Authority.

There are, of course, other powers: the power to require, to review, to ratify, to reject, to direct, etc. Each of these can be seen, however, as an exercise of authoring power: to state a requirement, to write a review, to issue a statement of ratification, to issue a policy directive, etc. While it may seem a bit artificial at first, construing authority as the power to create gives us a simple organizing principle for models that allows for a natural association between authorities (programs, projects, work packages) and the things they produce (designs, reports, hardware, software, science data, etc.).

With this in mind we can say that a primary role of the Work Breakdown Structure is *attribution*, ensuring that every decision made over the course of a project is traceable to a single authority—not directly to a person—but to a persistent authority that endures regardless of the assignment of any particular person(s) to it. Project Systems Engineering is a good example: every project has (according to the *JPL Standard Project WBS Template* [17]) a work package for Project Systems Engineering. This package has a more or less clearly-defined scope of authority, including authoring Level 2 requirements. Implicit in the allocation is the idea that the Level 2 requirements are those requirements *and only those requirements* attributable to Project Systems Engineering.

In a document-centric process attribution is implied by documents; Level 2 requirements are collected into a document and the entire document is attributed to Project Systems Engineering. In a more modern information-centric process the document is a generated artifact, constructed by querying a knowledge repository for L2 requirements attributable to Project Systems Engineering. The attribution in this case is explicit. (One of the key goals of information-centric practice is to supplement human cognition with computing; doing so requires making information explicit.)

The Work Breakdown View evolves as the project evolves. High-level architectural decisions in early phases determine not only what systems are to be acquired, but how work is to be delegated. A mission that commits exclusively early to solar power will not need to allocate resources or delegate design authority to engineering radioisotope thermal sources. The final WBS recapitulates the project's history of considering options, choosing, and delegating.

The uniqueness of a WBS to a project is at odds with the desire to standardize management and reporting practices across projects, collecting metrics, observing trends, etc. JPL has grappled with these opposing principles by publishing a guide to *Work Breakdown Structure Tailoring* [18]. The result is confusing, largely because of the overloading of the WBS number to indicate both type and instance. For example, the Spacecraft Work Package is required to have number "06". If the "spacecraft" is in fact a multi-module Flight System, it is permissible to number these modules "06A", "06B", "06C", etc., and then continue with the convention of designating their subsidiary packages "06.01", "06A.02", ..., "06B.01", etc. This in turn leads to absurdities such as multiple levels evident in the WBS diagram being considered the same level, gaps in numbering, etc.

A better approach is to respect the Laboratory's legitimate interest in standardization by defining a set of standard Work Package *types* (e.g., *Spacecraft*), delegation rules (e.g., *Spacecraft Systems Engineering* must be subsidiary to *Spacecraft*), and requiring each work package in a given project's WBS to explicitly instantiate exactly one of these types. Then any project can construct a WBS that matches its specifics, naming and numbering as it chooses, as long as it complies with the typing and composition rules. The Laboratory, on the other hand, could compare budget and schedule data across projects by querying for the *type* of information of interest, without regard to names and numbers.

Allocations

Of course, it requires more than mere authority to achieve many things. It may require time, money, access to facilities, etc. For this reason, delegation of authority is accompanied by allocations in the form of schedules, budgets, priorities, etc. These are the Concerns addressed by the Work Breakdown Viewpoint. A complete Work Breakdown View will associate with each Work Package (Authority) its specific responsibilities, that is, the things it is expected to achieve or produce, along with its allocations of resources. These specifics represent exercise of authority by the parent Work Package and are therefore attributable to it. The pattern repeats as necessary, with each Work Package authorizing subsidiary Work Packages with assignments and allocations. The recursion terminates when every leaf Work Package requires no further delegation.

Definitions

Prototype Elements	Properties	extends…
Authority		
Program		Authority
Project		Authority
Work Package		Authority
Supplied Element		
Element		Supplied Element

Relationships	Properties	participants
Authorizes		Authority authorizes Authority
Supplies		

■

A.3 "Level 4" "Engineering" Flight Subsystem Viewpoint

This "Level 4" "Engineering" Flight Subsystem Viewpoint is intended for application to JPL flight projects, which are those that develop and operate systems for missions in space. Flight projects possess one or more components that "fly", referred to here as flight systems, which are in turn comprised of the flight subsystems. This Viewpoint is intended to specify **realizational** Views for such subsystems.

Definitions

Level — Level designation is primarily a realizational consideration. By convention, a Level 4 component of a JPL system is a first-tier subsystem within a major, separately deployed, project Element. For instance, the power subsystem of a flight system would usually be considered a Level 4 component. The system comprising these subsystems is considered a Level 3 component (e.g., the flight system), while Level 5 components (e.g., a power distribution unit) comprise the subsystems, and so on. However, the criteria for this leveling has been comparatively flexible, resulting in typically uneven application across most systems, prompting formal definition a few years ago in Flight Hardware Hierarchy and Nomenclature [19]. These definitions remain recommendations though, not requirements; and furthermore, two distinct hierarchies are defined, where one finds "Assembly/Unit" below "Subsystem" in one and "Assemblies" above "Subsystem" in the other. Therefore, devising an appropriate leveling structure for realization should be considered part of the architecting effort, not driven solely by precedent or policy. With leveling established, one Level 4 realizational View would typically be assigned for each Level 4 subsystem, and each would be guided by this Viewpoint.

Various factors regarding functional coherence, tight integration, separable developments, collocation, fault containment, or other criteria will generally determine a suitable realizational breakdown. However, the functional units of this composition are commonly drawn from *conceptual* considerations, and from multiple points of view (e.g., packaging, thermal design, isolation and grounding, radiation shielding, etc.). Therefore, an important role for realizational Views at each level is to establish a workable implementation breakdown within which the various conceptual breakdowns can been reconciled. That is, **it is *not* the role of any realizational View to establish the functional decomposition of a system, at any level**. Rather, it is to perform a mapping of the conceptually established breakdowns onto realizable components. This is generally a bilateral, iterative compromise between conceptual and realizational interests, and similar issues apply across levels. In the case at hand, Level 4 Views can be developed only within the context of higher and lower levels. As a consequence, descriptions in this Viewpoint frequently refer to assertions established either realizationally at Level 3 or in concepts, as a way to explain the role of Level 4 and analogous concerns at that level.

Subsystem — In general usage, the term "subsystem" has at least three related but distinct meanings. In one sense, a subsystem is a work unit of a project's development effort, commonly associated with a particular work breakdown Element. In another sense, a subsystem is a functionally bound set of products that together comprise a major deliverable component of a system. As a rule, work unit and work product subsystems are paired one-for-one (ignoring historical oddities like "virtual" subsystems), because the functional grouping of flight system components and the assignment of responsibility for their collective delivery are deliberately linked as

a management strategy intended to ensure maximal coherence of oversight within key functional domains.

In the third sense, a subsystem is the subset of a system (i.e., a sub/system) identified with a particular function. Any item whose removal would render the function inoperable would be considered part of the subsystem, so specifications of subsystem functionality or performance necessarily entail all such items. However, this functional composition rarely aligns exactly with the work product subsystems noted above. Such overlaps are often close, but sometimes not close at all. For instance, the cables that connect power subsystem (second sense) components are involved in meeting power subsystem (third sense) requirements, and many others; yet the system harness is generally considered a separate subsystem (second sense).

Because of their apparent relationships, these three different meanings are often conflated to ill effect[3] by failing to distinguish between the entity responsible for performance, the entity assigned for development, and the entity that develops it. **This Viewpoint addresses subsystems in the *product* sense, with assertions directed primarily at a Level 4 entity under development**, invoking the developer sense only where choices of technical composition are influenced by work breakdown, and invoking the functional sense primarily as a way to achieve maximal design coherence within the chosen realizational decomposition.

To simplify reference to these different meanings, notation from here forward will be as follows:

- **SubSystem** (both initial and internal capitalization) will refer to the developing entity that is assigned delivery responsibility (sense 1).
- **Subsystem** or **subsystem** (no internal capitalization; initial capitalization by context) will refer to the delivered subsystem (sense 2), the focus of this Viewpoint.
- **Sub/system** or **sub/system** (internal slash; no internal capitalization; initial capitalization by context) will refer to a functional composition (sense 3).

Flight — Even the term "flight system" may be ambiguous. Some projects have chosen to apply this term to every part of the overall system that leaves Earth, including all stages of the launch stack. Others refer spacecraft, orbiter, probe, or otherwise, rather than flight system. Intentions here take the particular view that flight systems are those parts of the spacebound system that are developed or tailored exclusively to meet mission objectives. This could include kick stages, carriers, or other Elements that are abandoned at some point, depending on their specificity to the project, and it includes items that perform their intended functions only during in situ operations that involve no flying. Flight systems also spend a substantial part of their existence in ground-based testing before launch, during which they don't fly (except possibly during transport), but for which there are important architectural implications. There is no intent here, in using the term "flight", to exclude such considerations.

Engineering — For the purposes of this *example* Viewpoint, "engineering" flight subsystems (generally numbering around ten) are the focus of attention. However, from a strictly technical point of view, the engineering/science dichotomy is somewhat artificial, especially given Level 4 considerations. Some components are not cleanly one or the other, doing double duty for both

[3] One is likely to find requirements that begin "Subsystem X shall design…", suggesting that the design *work* is to be verified, but not the implemented design. Also, system-level performance requirements are often levied on subsystems (sense 2), even though the targeted subsystems cannot alone ensure conformance, due to dependencies on other subsystems. Such misdirection generally leads to a crisscrossed maze of laterally traced requirements.

engineering and science, and the distinction is even less apparent in systems dominated by one large instrument, which may significantly subsume "engineering" functionality. For in situ exploration, both science and engineering share mutual interests (for different reasons) in assessing the context of their operation; and so on. Thus, a general science instrument Viewpoint at this level would be quite similar to that presented here.

With such observations in mind, the aim of this "Level 4" "Engineering" Flight SubSystem Viewpoint is to address those aspects of *any* flight subsystem, instruments included, that are not exclusively about their ability to acquire science data. Therefore, this Viewpoint is necessarily non-specific regarding the differences that distinguish subsystems from one another. Instead, it addresses those issues that are often shared. It is evident, nonetheless, that broad classes of subsystems recur across most flight systems (power, telecom, attitude control, instruments, etc.), so **further detailed specializations of this Viewpoint for application to different subsystem classes are expected and *essential***.

Exclusions

Modularity — Besides partitioning into subsystems, other realizational partitions ordinarily arise in describing a flight system from a work product point of view, a common one being the notion of a *module*. In practice, a module is rarely comprised of a single subsystem, nor are subsystems necessarily part of just one module and no other. Instead, the criteria for module definition generally have more to do with issues of integration, where for convenience or necessity, it is advisable to stage integration into tightly integrated sub-units (i.e., modules) that may then be integrated into the final flight system.

Modularity can be motivated by either programmatic or technical considerations—often both. Programmatic modularity can be driven by split developments (e.g., partnerships or major procurement relationships); technical by articulation or re-separation in flight (e.g., scan platforms, atmospheric probes, rovers, etc.) or allowance for venue constraints (e.g., accommodation in special test environments). Whatever the reason though, modularity is a sufficiently unique issue to require a separate Viewpoint. Modularity is considered here only to the extent that it affects subsystem configuration.

Deployments — Over the course of its existence, a subsystem typically exists in other arrangements other than its final flight configuration, but where it must nonetheless satisfy at least a subset of the requirements imposed upon it. These arrangements are referred to here as *deployments*. A deployment may involve a complete subsystem in which only a subset of interfaces is engaged, or it may involve just a subset of the subsystem, where components are missing or replaced by alternate implementations (e.g., emulations). Both variations often apply in the same deployment.

Deployments can be defined for both flight and pre-flight situations. Modularity, as described above, is a possible motivation, but there are usually several others. Before flight, deployments typically occur during stages of integration and test, where a full system has yet to materialize. Despite their incompleteness though, such configurations are used for a substantial fraction of system V&V, so it is important that the tests performed in these configurations be relevant to the system as a whole, for which there is seldom time or capability to redo all such tests. As a consequence, subsystem requirements are likely to spill over into non-flight components and interfaces (e.g., emulators, direct access I/O, non-flight cables, etc.) or modes of operation (e.g., single

string operation, software-only testbeds, etc.) that are exclusive to these deployments. Sub-System Views address how this approach can be made plausible. Similar variants can occur during flight when separations, failures and other irreversible state changes, and other events alter the configuration in profound ways. In these cases, too, it is common (or should be, at any rate) for requirements to invoke particular deployments in their expression.

The necessity and specification of different deployments generally goes well beyond the exigencies of individual subsystems. Therefore, deployments deserve a separate Viewpoint and are considered here only to the extent that they affect subsystem configurations and the requirements that refer to them. It should be noted, however, that the extent of this dependency can be considerable.

Design and implementation — During the development of an architecture, many options are considered and eventually closed, as ideas converge on commitments for subsequent design and implementation. The resulting architectural assertions constrain the design, but they do not categorically determine it. Instead they leave ample space within which design can be exercised, just as design must leave allowance for variation in implementation. Therefore, this "Level 4" "Engineering" Flight SubSystem Viewpoint is directed primarily toward the assertions that establish the architecture within which design may progress. Design and implementation are considered here only to the extent that they affect subsystem architecture choices.

Restriction to architecturally dictated assertions applies even where cost, risk, or other considerations result in such limited options that only an already established product is plausible. In this case, while allowances must be made for heritage (effectively, another constraint), assertions should nonetheless express only what is essential to the integrity of the architecture, once reconciled with heritage, rather than merely reciting what the limited choice happens to be. Specific details are not appropriate to architectural Views guided by this Viewpoint. They may be addressed elsewhere, as needed, but the architecture description should focus on what is necessary for this heritage to be *acceptable*.

Similar exclusions apply regarding the assignment of implemented units to interchangeable placements across or within subsystem deployments. For instance, the SubSystem's sparing approach, selections for redundancy, diversions for test, and so on are not addressed by this Viewpoint, nor are considerations for prototypes, qualification units, or other such products. These topics may be addressed elsewhere, as needed.

Description

Scope — The scope of a subsystem View that is developed in accordance with this Viewpoint would generally be quite broad, if considered for its functional aspects. These are likely to include most of the following, and possibly a few others:

- Performance (range, resolution or sensitivity, accuracy, capacity, noise, error rate, nonlinearities, stability, alignment…)
- Redundancy (block/functional, voting, usage rules…)
- Ports or connections (power, data, mechanical/packaging, thermal, shielding, test, direct access…)
- Environmental tolerance (types, allowed ranges and stability, interference/disturbance to/from other components…)

- Electrical compatibility (grounding, isolation, interference…)
- Behavior (reset and startup, operating modes, configuration management, dynamics, I/O, warmup/cooldown, programmability, damage/waste/hazard avoidance, failure modes, fault detection, isolation, containment, and recovery…)
- Deployments (separation, release, extension, articulation…)
- Computing (computers, software, algorithms…)
- Data flow (storage, busses, networks, point to point connections…)
- Time (clocks, synchronization…)
- Reliability (failure rate, lifetime versus usage, environmental dependencies…)
- Resources (tracking and management of power, mass, data, propellant…)
- Geometry (shape, fields of view, configuration/emission space, orientation, reference frame, sense/actuation axes, phasing, alignment features…)
- Other accommodations (oversight, co-alignments, stray light…)
- Operational needs and constraints (maintenance, exercise, calibration, software or parameter updates, keep out zones, other "flight rules" …)
- Testing (simulated environment, emulators, monitoring and recording…)

All such topics can influence a subsystem View. However, it is important to note that for the most part, **such issues flow from conceptual considerations that are then mapped to the realizational subsystems addressed by this Viewpoint**. While a subsystem View may be productively organized in correspondence with conceptual divisions of this sort, **attention in subsystem Views should be to the mapping from these conceptual roots to the selected set of components that will realize them** as deliverable products. Except to the extent that conceptual constraints may be narrowed by realizational considerations, any added *functional* elaboration is probably better suited to conceptual Views.

As addressed further here, the mapping from concepts to realization is not merely a matter of declaring associations. Numerous rounds of development are involved as mapping proceeds, new components and interconnections are identified, constraints are negotiated and refined, delegation to lower levels, and so on. Each subsystem View that is guided by this Viewpoint would address this full spectrum of issues.

Composition — A Level 4 flight subsystem would have no purpose without a conceptual mandate. Concepts are vital in establishing the functional structures (composition and interconnections) that are embodied by subsystems. Nonetheless, the particular composition of any given subsystem, as a product, does not necessarily align entirely with any of its various conceptual counterparts (as discussed above). The mapping of conceptual Elements to realizational Elements is seldom determined by considerations regarding a single subsystem alone. Even where mapping choices are straightforward, or are guided by precedent, convention, or functional affinities, this mapping should never be taken for granted. Therefore, a key role of realizational Views involved in this mapping is to explain it.

Examples of mapping alternatives are easy to find:

- One might choose, for instance, to assign the load-bearing functions of an actively articulated joint to a mechanical structure subsystem, to which a separate controlling actuator belonging to another subsystem is then attached; or alternatively, both roles can be combined in a load-bearing actuator that is assigned to just one subsystem.

- A high gain antenna in a telecom subsystem might be chosen (or not) to supply sun shading so that a thermal subsystem need not supply a separate shade.
- Drive electronics for a propulsion subsystem engines and valves might be part of that subsystem, or part of a power subsystem, or part of an attitude control subsystem.

In these mappings, the choice is not just which sub/system Elements are assigned to which subsystems (as in the drive electronics example above). It may also be whether or not different sub/system Elements should be mapped to the same subsystem component (as in the antenna example), or which sub/system Elements should be split across multiple subsystem components (as in the actuator example). It is likely as well that some realizational Elements that would normally be considered a single unit must be partitioned in order to provide subsets that can be mapped cleanly to conceptual notions.

- For instance, it is common for fault containment regions (a conceptual notion) to straddle functional interfaces, such that the containment region encompassing one realizational unit also incorporates a part of another (e.g., the switch in a power distribution unit being within the fault containment region for the switched load).

In all such matters, subsystem assignments and cross-subsystem allocations are a Level 3 matter—not Level 4—unless the choice is explicitly delegated. This does not mean that Level 4 concerns are irrelevant to such choices. It means only that cross-subsystem considerations are to be addressed at the flight system level, given contributions from all concerned.

Situations like this highlight the distinction between subsystem as a developed item and SubSystem as a developer, where the difference between *who* and *what* is unavoidable and significant. Participation in the Level 3 mapping of flight system functional structure onto Level 4 components is clearly in a Level 4 SubSystem's interests. Indeed, SubSystems often author major portions of the conceptual Views that drive the subject mapping. Thus, their active involvement is essential, and the allowances they assert regarding what choices are acceptable are a vital part of any such trades. Level 4 Views should document such allowances. However, Level 3 is the deciding authority; and when choices are settled, those choices that matter at Level 3 are asserted and explained at Level 3. Only where options are overtly left open and delegated for lower level choice is Level 4 the driving authority, engaged correspondingly with Levels 5 and below (if defined). This pattern repeats across all levels.

A frequent variant of this pattern occurs whenever the existence of a Level 4 component is asserted for which there is no specific conceptual component to mandate it. This is most common where the *type* of component has been conceptually defined and accommodated, but without particular instances having been preidentified. Examples of this are also easy to find:

- Electrical power switches and their associated power loads are defined as part of a conceptual power sub/system, but without an a priori enumeration of particular switch↔load pairs. At Level 3, a choice to assign switches to the power subsystem and loads to other subsystems does not completely determine the composition of the power subsystem. Instead, whenever a component from another concept is realized and the realizing Element is declared to be a switched electrical power load, an associated switch must then be asserted to exist as part of the power subsystem.
- Given freedom to choose, a multicomponent subsystem (e.g., an instrument) might be integrated into different numbers of realizational units, depending on packaging choices, proximity constraints, or other considerations. A Level 4 choice to partition can lead to the need

for external connecting cables, which might then become part of a separate harness subsystem.

> This is an example of interface *reification*, where a sub/system concept establishes that components are to have an interface, but without asserting the realization of this interface. If an intermediary component (e.g., a cable) must be added to realize the interface, then the interface is said to be reified via this addition, and new interfaces arise as a consequence (in this case, box and cable connectors).

- The items (structure, chassis, optics, etc.) that determine mechanical alignment between two flight system components usually reside in different subsystems. Conceptually, the manner of assessing end-to-end alignment and sub-allocating an error budget may be defined, but the particular items in each alignment chain are determined elsewhere, wherever particular Level 4 components realizing the chain are identified.
- The thrusters in a propulsion subsystem reside within a conceptually defined propulsion sub/system concept, but constraints on their number and geometrical arrangement are likely drawn from concepts for attitude control, mechanical configuration, thermal design, and others. Thruster redundancy would reflect a fault containment concept, and so on. And that's just the thrusters. Level 3 mapping will address some of these issues, but unfolding details will prompt substantial expansion at Level 4.
- Similar situations arise with mounting structure and mechanical configuration, radiation shielding, pointing, inertial ballast, data management, and *many* others. The realizational components identified to fulfill functional roles are often complex elaborations of functional counterparts that have been defined only in general terms.

A key role of subsystem Views is to carry such additional realizational breakdown beyond what can be accomplished at Level 3 such that conceptual structures can be fully mapped. Where a new component is identified as an instance of some defined conceptual type, the associated conceptual View is liable to change as well with the introduction of additional constraints targeted to that instance, and the subsystem View can declare allowances that must be made for the added component. Thus, additions are subject to the same give and take between conceptual and realizational concerns as in initial rounds of mapping.

Whenever a Level 4 component is identified with *no particular* conceptual precursor (i.e., not just one of an already defined types), new Level 3 issues may arise with its discovery. Thus, it would be inappropriate to assume that every newly identified component is necessarily delivered as part of the subsystem where the need for it arose. Even a priori delegations, which are generally asserted by type (e.g., that all externally routed cables will be part of the harness subsystem) can allow exceptions (e.g., allowing connecting cables to be part of a partitioned instrument might be wise), so Level 3 always plays a controlling role.

> Whenever a need is asserted *before* any conceptual approach is defined to meeting it, this reversal prompts no change in eventual composition descriptions, once the need is addressed. However, unanticipated need may be sufficiently noteworthy to warrant explanation in a View.

Again, this pattern repeats across levels, so a Level 4 subsystem View should play a comparable role relative to subordinate levels (if any).

Interconnections — The assignment of flight system interfaces (and Relationships generally) among subsystems closely follows the pattern for composition described above. Most Relationships are defined conceptually among Elements, so the mapping of conceptual Elements to realizational components naturally translates these Relationships to subsystem components.

Authority for mapping choices therefore extends accordingly to Relationships. However, because each Relationship involves two or more components that may or may not be assigned to the same subsystem, it is not the Relationship itself that is assigned, but rather *participation* in the Relationship. This is proper, given that any given component can provide only their own contribution, which must be separately specified.

The authority for handling the details of design for any given Relationships lies at the level to which a Relationship has been mapped. For a flight system, Relationships that span multiple subsystems would be addressed at Level 3. However, it is also a Level 3 prerogative to assign components in a Relationship to the same subsystem, in which case this Level 4 responsibility is assigned by Level 3. In fact, the nature of Relationships is often a prominent factor in deciding the mapping to subsystems, because it becomes a SubSystem responsibility then to establish the details of design that realize subsystem Relationships.

Such matters may become tangled and confusing unless careful attention is given to the *conceptual* origins of Relationships, because different Relationships typically need to be addressed differently, even where the same components are involved. For instance:

- Given the flight system harness as a separate subsystem, a cable that reifies a conceptual, intra-subsystem data interface would typically be specified at Level 3, as necessary to address applicable flight system level concerns (e.g., routing, bundling, wire treatment, etc.). However, details of data content and protocol are normally irrelevant to a cable's design. These can still be specified at Level 4, because as far as the conceptual data interface is concerned, this is still an intra-subsystem interface, and the cable is not germane. Similar considerations would apply for network or bus interconnections among subsystem components.
- An inertial reference unit in an attitude control subsystem is typically attached to the flight system basebody in order that sensor rotations can be related to those of the basebody reference frame. Many smaller inertial bodies are rigidly connected to form the combined inertia of the basebody, and among the smaller inertial bodies is the inertial reference unit. Therefore, the mechanical interface to the inertial reference unit plays different roles in two distinct Relationships, one relative to the system reference frame, the other relative to the center mass. Both matter intrinsically to the attitude control sub/system concept, but neither is a Level 4 matter.

Upon a completely elaborated mapping of conceptual structure into realizable components, a full set of interfaces among subsystem components and with other subsystems is defined. These are the Relationships commonly represented in system and subsystem block diagrams or in interface lists and related artifacts. In a realizational setting, it may be appropriate to organize such information by component rather than by functional affinity. However, the particular realizational form taken by a conceptual sub/system is generally important as well, so both should be described.

Functionally, interfaces draw their purpose from conceptual origins, and each participant in an interface must support it with behavior in accordance with the concept in which it is defined. This would normally include performance criteria or other constraints on the interface properties of each participant that are significant to the chosen conceptual approach. The realizational implications of such constraints are to establish what ultimately become interface requirements on each component (sometimes captured in separate interface requirements document, *aka* IRDs, though they are just a subset of component-specific requirements). Subsystem Views must

acknowledge such constraints, but they do not define them. They may, however, further *refine* these constraints, as part of the elaboration of interfaces into realizable form. Such refinements contribute the interface requirement set for that component.

Further descriptive detail for shared aspects of an interface design (e.g., signal characteristics, data format definitions, fastener patterns, etc.) may also be included within a subsystem View, but it would generally not result in additional requirements beyond the component-directed constraints to adhere to such definitions.

Requirements — All constraints asserted by the conceptual architecture and mapped through levels to particular realizational products must eventually be solidified as Requirements. Each Requirement expresses an agreement between the parties involved. Within a work breakdown, requirement authority usually follows the work breakdown hierarchy, so at the subsystem level, requirements would reflect an acceptance by the SubSystem of the constraints levied on subsystem components by virtue of the mapping established at Level 3, for which the flight system work Element is the authority. Therefore, SubSystem acceptance of the requirements constitutes formal Level 4 acceptance of the mapping established at Level 3, and a corresponding Relationship applies at Level 5 with respect to the mapping performed at Level 4.

A potentially confusing aspect of this alignment between constraints and requirements is that the "flow-down" through levels via a work breakdown hierarchy might somehow be expected to mimic the functional elaboration that is explored in conceptual Views. It does not. Traceability of requirements in the conventional sense, as one requirement established to help fulfill another, derives *entirely* from conceptual considerations. What flows from level to level are choices about how to successively map functional roles onto realizable components. The consequent allocation of constraints is then embodied in a set of Requirements at each level. But traceability is about constraint dependencies, not requirement levels.[4]

Consequently, the collection of Requirements in a subsystem View should reflect authority flow and responsibility, comprising only those Requirements that can be met by the subsystem alone, and therefore that express commitments that the SubSystem can make unilaterally for the products it delivers. An expected product of this compilation is a Requirement document suitable for signature approval by the involved authorities.

Where Requirements can be narrowed even further to apply to a particular component within a subsystem, there is little value in associating the Requirement with the subsystem as well. Any issues with whether, or how, or where the Requirement is to apply belongs with the mapping process for aligning conceptual and realizational structure, which includes the mapping of constraints. An explanation of this mapping should be included in the subsystem View, but having done so, Requirements belong with the commitment to meeting them.

Assumptions — During the evolution of an architecture, before all constraints are identified, or while the specifics of constraints remain fluid, it is common for working assumptions to guide

[4] In conventional document-oriented processes, a typical manifestation of the distinction between level dependencies and functional dependencies is the appearance of "applicable documents" within a leveled document tree. Conceptual aspects of the product in question generally determine what is applicable. For instance, requirements applicable to a telecom subsystem component might address environmental design, electrical grounding, mass allocation, safety, fault tolerance, and several others that apply across many subsystems and at different levels. Flowing these through the level hierarchy is not necessary for traceability.

development with the understanding that these assumptions must eventually be resolved. Once an architecture is stable, there should be no open assumptions left. Therefore, if an architecture description portrayed merely this final state, it would have no place for assumptions. This is not the only purpose of an architecture description though. It is also an explanation for why the architecture is as it is. Solid rationale of this sort is essential to the stability of an architecture, because good decisions backed by solid reasoning are least likely to change, especially when everyone understand this rationale. Stability of the design space rendered in an architecture description enables subsequent design and development work to proceed smoothly and confidently.

The resolution of assumptions into actionable constraints is an important part of the process for bringing an architecture to stable closure. Assumptions are in effect questions that an architecture must answer. They generally arise from experience that suggests likely—or at least plausible—options that permit work to proceed at risk until these questions can be settled. Such assumptions can come in many guises. For instance:

- For entry into an atmosphere, we assume profiles for density, wind, and so on.
- For operation in the outer solar system, we assume that radioisotope power is necessary.
- For navigation to an asteroid, we assume radiometric services from the DSN are sufficient.
- Until instruments are selected, we assume a reasonable bound on data rates and volume.

etc.

Assumptions can apply to allowances as well. For instance:

- For mass estimates, we assume that cable mass can be extrapolated via a simple model.
- For pointing, we assume that star tracker performance will not be threatened in the destination environment.
- For actuator lifetime, we assume that prior qualification is applicable to the intended usage.
- etc.

Nothing should be taken for granted in such enumerations.

Resolution of an assumption generally takes one of two forms, though rarely one or the other exclusively. The first form is a demonstration of some sort, whether by observation, analysis, test, or expert evaluation that an assumption is correct. The other form is to accept that an assumption may *not* be correct, but that the likelihood of this is acceptable. Clearly, a project's interests lie in the first form, as much as possible. Much of what we do to gain confidence in the quality and reliability of a system before deployment is aimed at this outcome. The more solid the demonstration, the better. However, absolute certainty is a rarity (and prone to negation where it exists), and situations arise, especially where exploration is involved, that defy a priori resolution with negligible risk. Therefore, dealing with assumptions is part of the overall risk management process.

Models — As realizational decisions are made and the particulars of these choices are defined, it is generally necessary to consult such information in the conceptual analyses that assess realization relative to the constraint placed upon it.

Such information might be as simple as an estimated value for a property bounded by a concept, in which case margin between the estimate and the bound provides a measure of reassurance for the realizational approach. A rollup of realizational mass estimates, for instance, would be

compiled to assess both subsystem allocations and system mass margin overall, with comparison against asserted margin policy.

More generally though, what may be needed in order to characterize realizational choices are more elaborate models of composition or behavior. These complications often arise as a consequence of the convergence of conceptual variety in a single item. For instance, a device that requires power in order to perform its assigned function must provide a model that establishes this connection in order that scenarios used in conceptual analyses can take this dependency into account. Associated with this connection can be thermal constraints, startup modes, inrush power, interference with other items, competition for resources, and so on, each arising from different conceptual perspectives that all unite in the same realizational Element.

Whatever their form, complexity, or purpose, models of the consequences of realizational choices are part of every subsystem View. The models addressed a subsystem View are those that arise from the composition of subsystem Elements, while the models of mapped units *within* this composition are addressed at the next level. To the extent that a subsystem is well delineated in its functionality relative to other subsystems, the subsystem's model (or models) can be substantially different from a mere aggregation of unit models, and it is generally desirable that the details of unit models can be hidden or abstracted in a way that simplifies its usage at flight system or higher levels. This will depend on the topic of interest. For instance, an attitude control subsystem can generally be characterized by its combined pointing performance (ignoring added contributions from intermediate structure), but a model of power for the same subsystem would typically conceal little about the power of its components, except to provide modal bounds, which may or may not be good enough for system power management.

The models created for realizational description should be documented in accordance with architecting framework guidance for Models generally, declaring their nature, purpose, provenance, usage limitations, fidelity, and so on. The details of the Model itself will generally be in the form of constraints (on Properties, composition, or otherwise), and as such can be considered part of the allowances captured in every realizational View. That is, if a Model asserts some behavior or other characteristic, then all constraints levied on the modeled item must accommodate this possibility—with suitable margins.

Analyses — As realizational choices are asserted and the constraints associated with conceptual to realizational mapping converge in particular components, it becomes necessary to justify these choices by demonstrating that each component is realizable, as constrained, and that the collective choice for the set is well chosen among other alternatives that may have been possible. Demonstrating these attributes is an essential obligation for every subsystem View.

The rationale for these decisions will generally take the form of analyses in which the composition and properties of the subsystem are compared to the constraints mapped onto them. Models used in these analyses might be similar to those provided for conceptual analyses, but there are often added realizational considerations that can be confined to realizational consideration. For instance, a lifetime model might consider materials, temperatures, radiation, operating cycles, and so on that can be bundled into simple operational abstractions for conceptual interests. For this reason, realizability should be addressed differently in subsystem Views.

As with other models though, there may be both subsystem-level and component-level analyses. Subsystem Views should consider the realizability of their components in assessing the realiza-

bility of the aggregate subsystem, but the details for each component are better addressed at subordinate levels.

Description — Distinct from the constraints that define it and the models that describe it is a realizational Element's actual implementation. This is not an architectural matter directly, since a good architecture would deliberately make room for a variety of implementations. Nonetheless, documentation of the implementation provides an important service in validating the architecture. Wherever implementation struggles to meet an architecture or where the architecture must bend to accommodate reality, there are lessons to be learned, because the architecture has failed to establish a viable design space. Where a stable architecture permits smooth implementation, this too deserves attention, as an example to be emulated. A key purpose for rigorous architecting methodology is the continuity it provides from project to project and the legacy it leaves that enables steady improvement.

To meet these aims, the description of an implementation should include an overview of its design in a way that permits straightforward identification with conceptual representations and with the Models created for analyses in both conceptual and realizational contexts. The veracity of these Models, as part of the architecture description, is important. Likewise, a subsystem View should also report on or refer to the assessment of implementation against constraints imposed on the subsystem. These constraints are the source of requirements against the implemented product, so this inclusion creates explicit ties to the verification and validation efforts of the project.

Finally, the eventual flight history of a subsystem should also find a home in a subsystem View, as the ultimate validation of its integrity. This may not be possible until be many years after implementation, but when architecting is taken seriously, it is carried to the end.

Authorship — The scope of a View specified by this "Level 4" "Engineering" Flight Subsystem Viewpoint is a particular subsystem, as established at Level 3. A subsystem is typically assigned to a particular Element of the project's work breakdown (in this case, to a SubSystem), initially by discipline (for the most part) and eventually by the resulting composition, once this is determined. Accordingly, custody for the associated subsystem View would normally be assigned to the same SubSystem.

Where delegated from Level 3, a subsystem View asserts further choices regarding the mapping of conceptual structure to the components comprising the Level 4 subsystem. This View may in turn delegate mappings to Level 5, as deemed appropriate. These are SubSystem prerogatives, which it asserts and explains via the subsystem View.

As described previously, the mappings under consideration are from defined conceptual components to their realizational counterparts, so a realizational View is not responsible for expanding the conceptual component set to be realized (aside from problematic discoveries of unanticipated need, as already noted). However, an a priori enumeration of all realizational instances is rarely possible, generally awaiting the elaboration of functional components into realizable form. As this situation unfolds at Level 4, the expanded mapping is negotiated across levels, as with other components, with additions being asserted and explained in the subsystem View.

Through the mappings it declares, a Level 4 View serves as clearinghouse for the conceptual constraints allocated to a subsystem and its components. In addition, though, each such View declares any allowances deemed necessary to make the asserted mappings realizable, whether

from a technical or programmatic point of view. This is a key SubSystem responsibility, and a Level 4 View is the medium through which such allowances are reconciled with constraints mapped from Level 3, and then resolved into a workable Level 4 subsystem architecture.

Reporting Artifacts

To be provided

■

Appendix B View Templates

Following are View templates developed several years ago for the Europa Clipper project. They are well aligned with the architecting framework described here, but the circumstances of their development was not ideal and they were not extensively used, nor was adequate tooling in place at the time to support such use. Furthermore, the limited experience that *was* obtained with them demonstrated a tendency to treat templates as inflexible forms to be completed, rather than guidelines to suggest organization and content. Therefore, the notion of templates *per se* needs to be reconsidered. **With that caveat, these templates are offered as raw material from which future training material or other guidance might be derived.**

continued next page

B.1 Conceptual View Template

Outline

Introduction
- ***Issues***
- ***Assumptions***
- ***Context***
- ***Viewpoint***

Summary

Approach
- ***Composition***
- ***Behavior***
- ***Elements, etc.***
- ***Extracted Requirements***

Outcome

For complex Views, it may be useful to reiterate this template for each of multiple subtopics. That's fine; or you can distribute parts among sub-Views, wherever the parts stand-alone easily. A sub-View may have multiple parent Views, so a finer grain breakdown can help better organize related ideas.

Introduction

Issues

Indicate clearly the issues (from **Concerns** or **parent Views**) addressed by the View.

Issues should be either **success criteria** from Concerns, or **constraints** imposed in a parent View. The aim is to clearly show traceability from Concerns to Views, and Views to sub-Views.

Issues are a *subset* of all constraints defined within the scope of the View (i.e., within parent and higher Views and Concerns). In particular, they are the subset that has been assigned to this View for further elaboration. All other in-scope constraints are presumed to be true. However, any reliance upon these constraints should be noted explicitly in the View.

Assumptions

If additional **assumptions** need to be made, list *all* of them, clearly identified as assumptions. Each will be treated as an open item, to be promoted eventually (if accepted) to a constraint in a parent View or Concern, at which point the assumption can be dropped.

Context

List parent Views and/or Concerns from which the issues addressed in this View are drawn. Similarly, list sub-Views (if any) in which derived issues identified in this View are to be elaborated. This can be shown diagrammatically, if desired.

[Note: Lists will be accomplished automatically in later versions of View Editor via explicit links associated with issues. Similarly, ***context diagrams*** *will be created externally for transclusion into the View. The association rules described below will be prescribed.]*

The associations identified here are navigable in both directions. That is, a parent entry in a sub-View is necessarily listed as a child entry in the parent View, and vice versa. A sub-View can have more than one parent View, and conversely. An association between Views, or from Concern to View, is needed *only* where specific issues have been identified for elaboration, as described below under Approach. There are no lateral associations between Views. Overlaps are established by common parents and reconciled in shared sub-Views.

[Note: The association rules described above will be enforced automatically in the model.]

Viewpoint

Identify the Viewpoint (if any) that guides this View. If one does not already exist, make an entry noting this absence.

[Note: This will be accomplished in later versions of View Editor via explicit links. The absence of a link will be taken automatically as an open item.]

Standard nomenclature and representation formats, theoretical background, trusted data sources, institutional guidelines, helpful aids, and other information of this sort, which are likely to be applicable to multiple projects, are the proper subject of Viewpoints. These should be included by reference, where possible, rather than duplicating authoritative sources.

Summary

Explain briefly the overall approach to the issues at hand and an indication of how well they are addressed. This should be comparable in length and composition to the abstract for a paper. The purpose is to help the reader decide whether they've found information relevant to their interests. For instance, a list of summaries might be compiled as a concise directory into the architecture description.

Approach

The approach describes how the issues identified in the Introduction are to be handled. One should expect this section to comprise a major portion of the View.

It is not enough simply to claim that each issue is addressed satisfactorily. Such a statement requires justification. This is accomplished throughout the narrative of the approach by describing and explaining the architectural structure for addressing the issues of the View, and by providing supporting analysis, as necessary, to make a compelling case.

Justification should also include mention of the **Trades** (separately documented) that were performed to narrow options to the selected choice. They should also appeal to any analyses, tests, or other actions that demonstrate the effectiveness, efficiency, performance, or other characteristics of the approach pertinent to the issues addressed by the View.

Composition

Architectural structure identified in the approach will frequently include functional elaboration or augmentation of Elements identified in a parent View into a composition of interconnected parts (**sub-Elements**), showing how the parts work together to address the constraints imposed upon the whole. This explanation may unfold in stages, fostering new subordinate issues, which is why issues may flow to sub-Views for further elaboration.

Within such a composition, each sub-Element has an assigned role (i.e., its **Functions** and **Relationships**). One can think of these (and illustrate them accordingly) in terms of **block diagrams** (where boxes and lines are Elements and Relationships, respectively).

A typical system supports several such conceptual elaborations, according to what issue is being addressed. However, within any given View, focus should be strictly on the one composition that is relevant to the issues at hand, no matter how abstract and non-specific this might be. Thus, Elements needn't be particular system components, and Relationships needn't be particular system interfaces. This is why they are called *conceptual* Elements and Relationships. Details about such things, as needed to support meaningful analysis, are acquired through mapping to realization, as addressed in the Outcome section described below.

Sub-Elements and the Relationships between them will generally be subject to a set of **constraints** on their **Properties** that must be true in order to successfully address applicable issues. Constraints can express what we know or expect about items that already exist, or what we assert is necessary regarding the design or operation of items we make. Properties that require no constraints needn't be mentioned.

Tolerance for variation is a key aspect of risk management. Explain clearly what variations must be addressed in the architecture, whether driven by uncertainty or a need for flexibility. Describe how allowance is made in constraints for these variations.

Behavior

The approach addressed in a View will also generally need to describe intentions for using the items defined. This usage will exercise the behavioral characteristics of items, generally definable in **behavior constraints** on the time-varying Properties (i.e., states) of Elements and Relationships. These behavior constraints may be state equations, or they may merely be bounds on state variations. Since behaviors are commonly grouped by mode, behavior constraints can be expressed, at least in part, via **state machine diagrams**. Consult modelers for preferred conventions regarding the definition of Properties and associated behavior constraint expressions.

Some behaviors are sufficiently complex to require exceptional treatment. This can include expression in specialized tools, or modeling techniques needing extraordinary validation, etc. Such **Models** should be identified in the View as products in their own right that need to be separately documented and managed.

Often, a more elaborate description may be needed to fully describe intended usage. These cases are described in **Scenarios**, each of which lays out a progression of events or activities that unfold over time, involving the Elements cited in the approach as Scenario participants, and exercising the various behavioral characteristics that have been established in their behavior constraints. The objective of a Scenario is to show that the resulting aggregate behavior addresses the issues identified in the Introduction.

Scenarios can be expressed in a variety of ways, including **sequence diagrams**, **activity diagrams**, and others. Consult modelers for preferred conventions regarding Scenario Representation.

Conceptual Scenarios address *only* those aspects of usage that are directly related to the issues at hand in the conceptual View. Details about the interrelationships among various conceptual Views are addressed separately in integrated, realizational Scenarios, where the con-

straints describing crosscutting behavioral dependencies are addressed. Conceptual Views can refer to these realizational results in their assessments of an approach.

[Note: Scenarios will be separately documented and mentioned by reference at some point, in order to permit Views to exercise shared Scenarios. This will generally not be suitable for conceptual Scenarios (given separation of Concerns), so for now, documenting conceptual Scenarios as part of the View is appropriate.]

Elements, etc.

With the approach complete, list the Elements that have been added and exercised in Scenarios, and for each identified Element, list the Properties and Functions assigned to it. In addition, list the identified Relationships and their Properties. The preceding narrative, including all constraints articulated within it, should be expressed solely in terms of these listed items.

[Note: These listings will be accomplished automatically in later versions of View Editor by tagging terms within the text. These items will then be linked to one another, wherever they appear, including in block diagrams, tables, and so on, such that all instances remain consistent.]

Extracted Requirements

Upon mapping conceptual Elements to realizational Elements, the assertions made in the approach become requirements, as follows:

Implementation is required for each realized Element (and by association, its Relationships, Functions, and Properties).

Satisfaction is required for each derived constraint (behavioral or otherwise) on the Properties of Elements and Relationships.

Operation is required to follow the manner described in Scenarios for participating Elements.

Each such requirement should be listed. It will be a common pattern to work backward from requirements to architectural assertions in Views, but the objective in the long run is to make requirements to result of View assertions, rather than their source.

[Note: This requirement listing will be accomplished in later versions of View Editor by tagging terms within the text. Assertions will then be extracted and re-expressed automatically in requirement form, subject to further refinement by the user.]

Outcome

Revisit the issues, stating how well they are addressed by the approach, given their full expression in realization. That is, for each realization of a conceptual element defined in this View, apply the patterns of composition and interaction defined in this View, and show how the issues addressed in this View are satisfied in each case. Sub-allocations, as additional derived constraints, may be warranted.

Each issue identified in the Introduction should be handled individually and explicitly across all realizations. Where the issue is expressed as a constraint, the results of analyses should be cited, indicating expectations relative to the constraint. This would typically include an assessment of uncertainties and the margins put in place to accommodate these potential variations. Where outcomes are particularly sensitive to certain factors, even if those factors are well constrained, this feature should be highlighted.

The analyses used to support conclusions will sometimes be sufficiently complex to require exceptional treatment. This can include assessment in specialized tools, or analysis techniques needing extraordinary validation, etc. Such **Analyses** should be identified in the View as products in their own right that need to be separately documented and managed.

Indicate the traceability path for constraints gathered from realization that are subsequently sub-allocated in Outcome.

□

B.2 Realizational View Template

Outline

Definition
Realizational Element(s)
Conceptual Mapping
Categorization
Mapped Constraints
Generic Constraints
Specific Constraints
Assumptions
Requirements
Description
Allowance Constraints
Integrated Scenarios

For complex Views, it may be useful to reiterate this template for each of multiple subtopics. You can also distribute parts among sub-Views, wherever the parts stand-alone easily. In the following discussion, any references to "separate realizational Views" can be taken to mean either of these two organizations. However, unlike conceptual sub-Views, where the View hierarchy reflects the interwoven flows of functional decomposition, a realizational sub-View may *not* have multiple parent Views. The realizational View hierarchy is strictly organizational, in the manner of a document tree. It should be comparatively flat.

▼ If this template must be **abridged** for some reason, non-heading paragraphs preceded by '**▼**' (as in this instance) may be omitted, along with immediately subsequent non-heading paragraphs at the same or increased indentation level. Endnotes are also optional. However, any abridged result should be clearly labeled as abridged and should include a reference to the full template.

▼ Only paragraphs beginning with an **imperative** (Indicate… List… Describe… and so on) are intended to invoke realizational View content. Everything else is provided as explanation and guidance.

▼ In all instances within this template, sections specifying a ***list*** should be constructible *automatically* from the system model, once the appropriate model connections are made. Every subsection of this template with an *Italicized Heading* consists entirely and exclusively of such a list. Guidelines here for these lists are intended mainly to facilitate the

identification of appropriate modeling connections, *not* to prompt additional documentation for the identified items except where ***additions*** are explicitly noted.

Definition

The essential purpose of a realizational View is to gather all relevant information about a particular deliverable product within the architected system, or about a related item in an external system or environment, or about some meaningful composition of such things. These are referred to as realizational Elements.

The essential character of a realizational Element is established via its identification with various conceptual Elements. A conceptual Element is merely a restrictive *representation* of some realizational Element; so conversely, a realizational Element *is* each conceptual Element that it realizes. The distinction between them is entirely in the information exposed, which in conceptual Views is topic-specific, and in realizational Views is product-specific (or item-specific for non-products).

▼ Defining realizational Elements is a fundamental aspect of system architecture, because these Elements constitute the system to be implemented in accordance with the architecture. Substantial deliberation and review are necessary to settle on an appropriate collection of realizational Elements, properly categorized and mapped, so **proper Definition, as follows, is an essential first step** for any realizational View.

Realizational Element(s)

Indicate clearly what **realizational Element(s)** is addressed by the View. In this context, to be *realized* means to be given *actual or physical form* through an explicit mapping of one or more conceptual Elements to a designated realizational Element. Thus, realizational (i.e., actual) Elements are the concrete forms taken by conceptual Elements asserted in the architecture.

▼ To put it another way, conceptual Elements are the abstracted representations of real system Elements, as considered from some conceptual point of view (i.e., from the perspective of an operator, a test engineer, a product manager, a designer in some discipline, a stakeholder for some concern, an environmental specialist, and so on). In this regard, it is as possible to start with something real and treat it from different conceptual points of view, as it is to see different conceptual points of view converge in some real thing. Therefore, the mapping between concept and realization is not to be considered sequential, as in concepts preceding realizations. For some things, concepts come first, and reality evolves (as in design); for others, reality comes first, and is then abstracted (as in science); and a little of each can happen too (as with legacy technology). From an architectural point of view, conceptual and realizational Views are peers, each needing to be reconciled with the other.

As peers, the concept–realization mapping is navigable in both directions. That is, a realizational View must be cognizant of all conceptual Elements that are realized by its realizational Element; and conversely, a conceptual View must be cognizant of all realizational Elements that realize its conceptual Elements. This mapping is consequently a central means (the other being by top down elaboration) by which lateral architectural connections are made and mutual consistency is established.

An essential feature of a good architecture is tolerance to variation, whether such variation arises from acquisition choice, development uncertainties, manufacturing tolerances, measures taken for decoupling, margin, or other reasons. Therefore, when a realizational View is about a product, it is generally ***not* about a particular design or implementation**.[1] Rather, the intent is to reflect and embrace the range of possible products (different designs, parameters, configurations, providers, or whatever) that are declared acceptable within the conceptual architecture. A realizational View narrows what the conceptual Views allow *only* to the extent reflected in conceptual mapping choices.

The same principle applies when a realizational View is about a specific, preexisting, external item, which may nonetheless be subject to uncertainty or change. Whether this is another system or an environmental item, it is still necessary to characterize the range of possibilities that must be accommodated or tolerated.

Each realizational View is generally dedicated to ***one* particular *kind* of Element**, all instances of which share a common description. Distinct kinds of Elements would be described in separate realizational Views. The emphasis on *kind* here is important, because the realizational Element described in a realizational View may be instantiated in multiple delivered units of that kind.[2]

Conceptual Mapping

List the conceptual Elements that are realized by the Element described in this View.

▼ A realizational Element will typically be the realization of several conceptual Elements from different conceptual Views, each with unique properties and behaviors to assert.[3] That is, a realizational Element can be considered from many conceptual points of view. All must be considered. A major role for realizational Views is to provide a centralized accounting for such assignments as a means of promoting consideration of all relevant issues for a given item.

Where a conceptual View defines a number of conceptual Elements of one given kind, there may or may not be more than one realizational Element filling this role (with corresponding realizational View), depending on the nature of this plurality, as follows.

Interchangeable Elements: If a set of conceptual Elements is defined where members are asserted to be transposable, then only **one shared kind of realizational Element** must be defined in the realizational View. In cases like this, an ***a priori* conceptual enumeration** of distinct roles will typically have been asserted for these parts, leading to their **differentiation by role** within a composition rather than by individual specialization. Differentiation of roles is thus accomplished through properties and behaviors of the composition (or its Relationships), not properties and behaviors of the Elements comprising them.[4]

Different but similar Elements: If a set of conceptual Elements, all of the same kind, is defined in a conceptual View, but that View does not assert an *a priori* enumeration of them, then any Element that is declared to be a realization of that kind of conceptual Element thereby becomes a member of the set of conceptual Elements of that kind.[5] Such cases rely upon **enumeration by realization** in order to identify all instances.[6] A **separate realizational Element for each case** must be defined, and these will usually need to be defined in separate realizational Views. While all share the characteristics of their kind, these separate realizations (to the extent the conceptual View al-

lows) can be individually tailored to their application. **Differentiation is by specialization**, so these realizations — described separately — are not interchangeable.

Categorization

Indicate broadly the sort of realizational Element(s) under consideration. This categorization is mainly an organizational tag, helping to guide the sort of information that is relevant to a particular item, and to differentiate meta-level architectural considerations, which can become confused.

▼ From an architectural point of view, a realizational Element will frequently be a **terminal Element**, where no assertions have been necessary regarding the internal composition of the item and where all features of interest are expressed at or behind its interfaces.[7] Each such Element will typically appear within its own realizational View, dedicated to the consolidated description of that Element.

Either a terminal realizational Element is an item to be delivered as part of the architected system, or it is a part of some other system comprising the context of the architected system. The former are deliverable **products**[8] that are subject to constraint by the architecture, which specifies their existence, purpose, relationships, composition, usage, and so on. The latter may be either **external capabilities**[9] with which the architected system has negotiated dependencies, or **environmental features**[10] that the architected system must deal with in some manner.

These may be further categorized as **technical Elements**, comprising engineered components (e.g., a hardware assembly), operational teams (e.g., a test team), and the context within which they operate (e.g., a launch vehicle, a planet, or a radiation field), or as **programmatic Elements**, comprising project components (e.g., management and work breakdown structures) and their collaborators (e.g., development organizations, facilities, or services).

A realizational Element can also be **a composite Element** (i.e., composed of other realizational Elements). The constituents of a composite realizational Element are frequently of the same category, but mixtures are also common.[11]

It is likely that each constituent in this case would already be described separately in its own realizational View, while architecturally relevant Relationships among the constituents are acknowledged or established in conceptual Views. Therefore, the only novel statements that a realizational View can make regarding a composite realizational Element are either the result of aggregating information from the constituents,[12] and/or of disambiguating conceptual compositions or Relationships that have been deliberately left open by the concepts.[13]

Mapped Constraints

Constraints that apply to a realizational Element come in different flavors according to their origin, specificity, and other characteristics.

▼ Many constraints originate in **conceptual** Views:

A conceptual constraint can be **generic** or **specific**, according to whether it is applied broadly or is tailored (i.e., specialized) to each Element.

A specific conceptual constraint can be **isolated to just one** realizational Element or **involve others** in the broader context within which the Element resides.

A specific conceptual constraint can arise from ***a priori*** considerations motivated by the Approach, or as the result of **case-by-case** considerations in the Outcome for each identified instance.

Other constraints originate in **realizational** Views:

Assumptions are constraints that are expected from conceptual Views, but that have not yet been levied. Upon adoption by a conceptual View, they are no longer assumptions.

Allowance constraints reflect architectural assertions regarding the practicalities of realization, around which conceptual constraints must conform.

Except for the last, which is addressed below under *Allowance Constraints*, these are handled, as follows.

Generic Constraints

List all constraints acquired by the realizational Element solely by virtue of its concept–realization **mappings to prototypical conceptual Elements** (i.e., constraints that apply to all Elements of a given kind).[14]

▼ Generic constraints are subject to the same acceptance process that applies between successive conceptual Views.

There should be nothing to add here that's not already addressed in the originating conceptual Views. Associated case-by-case supplements[15] are possible, but where needed, these are handled as *Specific Constraints* (below).

The Description (below) should reveal how generic constraints are to be addressed.

Specific Constraints

List all constraints associated with the realizational Element by virtue of any **specific reference from a conceptual View** to that *particular* Element. In some cases, particular Elements are defined in conceptual Views and are therefore immediately subject to specific constraints. Where particular instances arise subsequently through conceptual mapping, the effect of a specific constraint is to *specialize* an Element with respect to general characteristics identified by the conceptual View.[16]

▼ Here as well, specific constraints are subject to the same acceptance process that applies between successive conceptual Views.

Constraints that specifically refer to the properties of one particular Element may nonetheless involve other Elements as well. Three cases are of note.

If a constraint involves **two or more *particular* Elements**, then the constraint is really about some composition. Therefore, it should be dealt with in some other View that addresses the realizational Element embodying their composition.[17] Therefore, constraints for this case **should *not* be included**, because they will be listed instead with the composite Element.

If a constraint involves ***one particular*** **Element of the architected system plus other unspecified Elements** of some indicated kind (e.g., in some conceptually established pattern) **and/or other Elements (particular or not) of external systems or environments**, then the constraint is of motivating relevance only to the particular Element. It should be listed, even though the constraint has broader implications.[18,19] These are effectively a set of **accommodation constraints** for the realizational Element, so must necessarily be a matter for attention only in some conceptual View other than the one that originated it (refer to the note below regarding the Accommodation Pattern). Because the accommodation is handled elsewhere, the Description (below) need not address it.

Otherwise, **only one Element** is involved (though it may be either composite or at a terminal level). Clearly, any specific constraint isolated to this Element should be listed. Of these three cases, only this case directly constitutes a requirement on this particular realizational Element. The Description (below) should reveal how such specific constraints are to be addressed.

There are two situations where a conceptual View would specifically indicate an Element that is uniquely realized by one particular kind of realizational Element. Their mention depends on whether or not they arise from an *a priori* enumeration by the conceptual View, as follows.

An ***a priori*** **enumeration** occurs wherever a conceptual View defines a few conceptual Elements that are unique to the functionality addressed by the View and that are defined specifically to serve some purpose within this context. Often just one of each such Element is needed in the conceptual composition, but sometimes collections of interchangeable Elements are needed, which work together in a prescribed way.[20] In these cases, each kind of conceptual Element is mapped to one realizational Element. Therefore, any constraint levied in the conceptual View applies specifically to a unique counterpart in some realizational View.

The alternative is **enumeration by realization**, where in many conceptual Views, the existence of (or need for) additional conceptual Elements can be acknowledged, but no *a priori* enumeration by the View is appropriate.[21] Instead, enumeration occurs through the concept–realization mapping that indicates each specific instance, thereby identifying them as Elements relevant for consideration (typically within the Outcome section of a conceptual View).

Because enumeration by realization makes allowance for these instances to be different from one another, despite their similarity in kind, it may be appropriate for a conceptual View to consider each such instance on a case-by-case basis. In doing so, the need would be determined for further constraints that are particular to each case and thus directed to specific realizational Elements. These specific references often arise from a need to sub-allocate a shared resource,[22] or to sub-allocate contributions to a shared performance budget.[23] However, other features that are particular to each realizational case are also potentially subject to such added constraints. Likewise, although these added constraints are likely to be of the same sort across all cases, varying only by a particular constraint parameter, variations that are more complex are also possible.

The Accommodation Pattern

It is vital to note that ***specific constraints on a realizational Element do not necessarily mark the terminus of constraint elaboration***. In fact, since iterative reconciliation across all the lateral interdependencies within an architecture is a defining characteristic of most architecting efforts, these specific constraints frequently mark just one step in the multi-step process of finding, propagating, and converging a broadly interconnected set of constraints. It is preferable, therefore, to consider realizational Views, not as end states in a flow-down evolution, but rather as hubs for the interaction of concepts, where their competing interests can be resolved.

The general pattern for this interaction (*not* part of the realizational View) is as follows.

To start, suppose it has been determined that some Element is of interest to a conceptual View and that it is subject to a specific constraint. The View may have established the need for this Element in the first place, placing specific constraints upon it,[24] or the Element may be one of a set of relevant Elements enumerated by realization and constrained elsewhere.[25] In either case though, we suppose this Element to be involved in a constraint that governs properties within the ambit of the conceptual View.

If the specific constraint is exclusively on or among properties of one (possibly composite) Element alone, then there's likely to be little if anything to add, and this thread of interactions has reached its end. So, suppose instead that the constraint involves some Relationship with additional unspecified and/or external Elements (as described above). To proceed in that case, it would be necessary to explicitly enumerate the unspecified Elements concerned with the mutual constraint, according to which realizational Elements match the criteria of the constraint.[26] [To be clear, the conceptual View need not *assert* the matching Elements, but it does depend on what they are, as expressed in *Allowance Constraints*, discussed below.]

From the mutual constraint, there is now an opportunity to separately derive additional individual constraints on the matching Elements, from which compliance with the mutual constraint would ensue.[27] Each of these derived constraints is a specific constraint associated with a particular realizational Element, so the process is reiterated, broadening at each step to more realizational Elements.

In some situations, rather than (or perhaps, in addition to) deriving constraints on each of the separate Elements sharing a mutual constraint, it is possible instead to constrain properties of the realizational composition within which these Elements reside, such that the mutual constraint is met. The effect of this is to declare a realizational composition that is intended to address the mutual constraint (in the same way that one might declare particular properties for some Element at the terminal level). For a composition, this would be asserted through properties of internal Relationships (rather than through the properties of a constituent Element).[28]

Similar considerations can apply to dependencies *within* a realizational Element, where a mutual constraint among different conceptual Elements falls not upon a realizational composition, but upon a single realizational Element to which multiple conceptual Elements have been mapped.[29]

Iterations stop when no further specific constraints need be derived.

Of course, any flow down of issues in the conceptual View hierarchy that declares an *a priori* division of responsibilities and allocations largely preempts this accommodation pattern. Whether to address an issue in this top-down manner or to reconcile through lateral accommodation is a key architectural choice.

Assumptions

List all additional assumptions, as necessary to suggest potentially missing assertions from conceptual Views, clearly identifying them as assumptions and documenting each according to the presumed need.

▼ Assumptions are potential constraints. Each assumption will be treated as an **open item**, to be promoted eventually (if accepted) to a constraint from an associated conceptual View, at which point the assumption can be dropped.

Upon consideration and acceptance, an assumption identified specifically for a particular realizational Element may ultimately be determined to apply more broadly.

Note that "self-imposed" constraints are also a possibility, where a realizational Element is constrained for reasons not motivated or accepted by the conceptual architecture. If these persist and do not otherwise result in unreconciled conflicts, they can be retained, but should be treated merely as preferences or expectations.

Requirements

List every constraint (or conjunction of overlapping constraints, defined below) on this realizational Element that must be treated as a requirement because all of the following conditions apply:

1) The realizational Element is a **product** within the architected system.
2) The constraint originates in a **conceptual** View (e.g., *not* an assumption).
3) It is either a **specific** constraint exclusively on this realizational Element, or it is a generic constraint applied to this realizational Element by virtue of an **explicit** concept–realization mapping.
4) The constraint asserts **no dependencies** between the properties of this realizational Element and the properties of disjoint Elements (i.e., other than component Elements, if composite).

▼ Clearly, requirements are merely **a subset** of the collection of constraints associated with a realizational Element (as listed above under *Generic Constraints* and *Specific Constraints*).

In this context, **overlapping constraints** are those that constrain the same properties of the same Element.[30] These constraints would generally arise in different conceptual Views for different reasons, so they would normally not constrain the properties in the same way or to the same extent. However, since all must be satisfied, their intersection becomes the operative constraint. Only one requirement statement is needed to encompass this intersection, but traceability back to each of the contributing constraints must be retained.

Add a "shall statement" rendering of each listed item.

▼ Expression of such constraints (or conjunctions of overlapping constraints) in the standard "shall statement" format for requirements is necessary only when designation as a specified product is assigned. This designation is indicated in the Categorization (described above), but it does not become formal until the Element is associated with a delivering work breakdown Element, at which point the requirement becomes a binding criterion for the product upon delivery from the work breakdown Element. Similar considera-

tions apply for the authorization of product requirements, as part of a globally applied architectural pattern for assigning programmatic responsibilities for products. Thus, the notion of a "requirement" is not formally part of the architecting framework. Its mention here reflects an expectation for this standard, prominent application pattern.

Constraints on external systems or environmental Elements are not requirements, because their scope is outside the architected system. Instead, they are merely presumed to be true by virtue of agreement (for external systems) or consensus (for environmental Elements). Where the range of possibilities is uncertain, these constraints can be used to declare the bounds that will be accepted across the architecting effort for the purpose of uniform risk management.

Description

Provide a general description of the item at a level appropriate to addressing the conceptual architecture, and explain the essential characteristics that embody or enable this realization.

An elaborate exposition here is not necessary. For heritage or off-the-shelf implementations, it might be as simple as itemizing the options that are capable of meeting needs, along with appropriate references and a brief justification of plausibility. For significant adaptations or new designs, more information is welcome. In either case though, the aim is to explain what the realizational Element is well enough to defend the conceptual mapping and to explain and justify any Allowance Constraints, as listed below.

Differences in the nature of various realizational Elements will motivate adaptation in the nature of content within realizational Views, as follows.

For a **realizational Element at a terminal level**, description is directed to the intrinsic characteristics of the Element. However, these vary according to the Categorization of the Element, as described above.

▼ If the subject realizational Element of this View is a terminal **product** (i.e., there is no need, so far, to assert further architectural decomposition),[31] then summarize relevant features of the product. Describe (as appropriate) options for achieving the functionality it is assigned by the associated conceptual Elements, and describe the resulting implementation considerations that are relevant to its ability to realize these capabilities. These could include physical or practical limitations of available technology, the range of available vendor offerings or heritage designs, resource or other support needs, other coupling characteristics that bind otherwise separate behaviors together, or any other aspects that might present programmatic difficulties or competing technical factors in meeting constraints acquired from realized concepts.

If, from an architectural point of view, the subject realizational Element of this View is a terminal part of some **external system** (i.e., there's no need to be concerned with further internal structure),[32] then describe the presumed capabilities upon which we will rely. These would typically include performance capabilities or interface characteristics, but not internal structure. If there are reciprocal constraints back upon the architected system as necessitated by utilization of an external system, these are not part of the external system's description, but rather are assigned to the architected system by whatever concept invokes the external capability in the first place.

If the subject realizational Element of this View is a terminal part of the **environment** around the architected system (i.e., no further environmental partitioning is relevant),[33] then describe the presumed environmental characteristics with which the system must be compatible. These will typically be selected design ranges for the phenomena, chosen for acceptable risk, rather than the far extremes of physical plausibility. Therefore, the description should offer some justification for the selected ranges. This description should *not* address the manner or degree of interaction between the architected system and the environmental Element. These are addressed instead in whatever conceptual View deals with the architectural implications of such environmental effects.

Note that products and external system can also be environmental Elements.[34]

If the subject of the View is a **composite realizational Element**, description is directed at integrated characteristics of the composition, the constituents having been described individually in their own Views. What's relevant will depend, as before, on the nature of the composition, but also on the sort of Relationships that connect the constituent Elements.

▼ If the composite realizational Element addressed by this View is composed of **interacting realizational Elements** (described in other realizational Views) whose behavior is governed by or contingent upon effects among the parts, then describe the integrated "subsystem" that these comprise.[35] In most cases, it will be unnecessary to reiterate functional aspects of this integration (i.e., which parts do what for whom), since these will have been defined by the concept from which the composition is drawn in the first place. Therefore, the description should attend instead to remaining ambiguities regarding associations,[36] layout,[37] or other features of the composition that are architecturally relevant, but not derivable separately from the parts or their conceptual origins.[38]

If the composite realizational Element of this View is simply composed of **associated realizational Elements** that are of the same kind or in a shared Relationship, but which have no other interdependencies that are relevant to the collection (they *may* be interdependent in other respects), then describe the relevant aggregate features of the collection.[39] In most cases, it will be unnecessary to reiterate the reason for addressing particular features, since these will have been defined by the concept that declares their relevance.

Both cases may apply for some composite realizational Elements, where the parts interact, but only by virtue of some mutual association. Dynamically negotiated resources are a common example of this, where otherwise unrelated components that have been associated with the same resource are thereby forced by competition into interdependent behaviors.[40]

Note as well that different compositions can be of the same kind, in the sense that they share a common compositional pattern.[41] Each instance of this pattern is effectively a specialization of the pattern, in a manner analogous to the specialization of Element kinds at the terminal level. Recurring patterns are essential to architectural integrity, because they are the basis for understanding integrated system capabilities and behavior.

These compositions will sometimes comprise combinations of Elements both internal and external to the architecture. For instance, an interface with an external supporting system is a Relationship between two parts of a composition that contains both internal and ex-

ternal Elements. This applies to composition with external environmental Elements as well.

It may occasionally be necessary to break a terminal realizational Element into parts in order to find an appropriate concept–realization mapping.[42] In that case, it becomes a composite realizational Element instead, and each of its newly identified parts occupies the terminal level.

In the course of describing a realizational Element, the manner in which **conceptual constraints** are to be met should be addressed. Therefore, scattered references within the narrative to these constraints (as listed above) are expected, such that the implications for each constraint are clear. (Note, as mentioned above, that the Description need address only those constraints that are exclusive to this realizational Element.

Similarly, practical limits or unavoidable dependencies that are identified in the Description will prompt the creation of **allowance constraints** (described below), which the set of imposed constraints must accommodate. These too should be referred to in the narrative, as they are introduced.

Allowance Constraints

List and explain any further constraints that are necessary to allow for the reality of casting this Element in a realizable form.

▼ These added realizational features are most often needed for products (and sometimes external systems) in order to reconcile the practical consequences of integrating diverse conceptual characteristics within one actual implementation.

The aim here is to go beyond the general Description (as above) with verifiable characteristics that complement the constraints acquired through the concept–realization mapping (as addressed under Mapped Constraints above).

Realizational features are expressed through **added constraints** on or among conceptually defined properties (or for composite realizational Elements, on their internal Relationships). These properties can include state variables; so added constraints can address both **static and dynamic** (i.e., **behavioral**) dependencies.[43]

Such features should not normally involve new properties. Adding properties should be avoided where possible, because exceptions can be problematic, given no conceptual basis for dealing with them.[44] However, **extended value domains** for these properties are often necessary.[45] Added properties or extended value domains will prompt specialization of the realizational Element definition.

Allowance constraints will sometimes express **additional accommodation needs** — in this case by virtue of a realizational assertion that certain choices must remain unconstrained. This stipulation might reflect immaturity in design, uncertainty associated with unresolved questions, options held open for delayed choice (e.g., in competitive procurements), implementation Trades that have not yet closed, and the like; but they can also reflect the realities of limited choices, as for instance when available heritage options are meager. Whether to tolerate variation or simply to acknowledge limited options, conceptual views must not impose limit constraints that encroach upon such actualities, so allowance constraints are established to avoid this.

An allowance constraint can also express a narrowing of options that is not forced by practical limits or unavoidable dependencies, but rather that reflects the firm resolution of some choice. These **design commitments** have a similar character to the actualities described above, in the sense that some possibility needs to be protected. The reasons in this case, however, tend to deal more with the practicalities of maturing a design than with intrinsic limitations.[46] The typical problem here is a frequent necessity to narrow the scope of analyses to more particular cases. Having done so, it isn't necessarily desirable to limit design *only* to these choices (effectively point designs); if different designs can still meet requirements, they should *also* be permitted. Nonetheless, having devoted time and effort to the analysis of some variant within this design space, it is prudent, past a certain point of maturity, to protect this investment, just as one might protect practicable implementation choices. By reserving some choice (or perhaps a range of choices) and capturing it in an allowance constraint, a guard is put in place as protection against subsequent changes to imposed constraints.

Note that allowance constraints do not meet the criteria above for ***Requirements***. In principle, these constraints can change, as long as requirements are still met. Nonetheless, because they establish assertions upon which conceptual analyses are based, their change should be managed cautiously and their veracity should be subject to verification. In these respects then, they carry similar weight to requirements.

The logical nature of a constraint that makes allowances is distinct from that of a constraint that imposes limits (as described under Mapped Constraints).

Limit constraints narrow choices, making the set of acceptable systems smaller, while allowance constraints preserve choices, ensuring that the set of acceptable systems is not too small. Each additional limit constraint can *reduce* the possibilities that will be accepted. Each additional allowance constraint can *increase* the possibilities that must be accepted.

This is a vital distinction, even though the expression of actualities and commitments in allowance constraints can seem quite similar to imposed limit constraints. The reason becomes clear if allowances are inappropriately treated as fact in analyses or in Trades. Architectural choices can then be made on the presumption of a particular design (e.g., a point design), even though limit constraints describe a larger set of possibilities. As a result, tolerance to variation is sacrificed — often with unpleasant consequences.

The *proper* approach is to base all analyses, Trades, etc., as practicable, on the full range of possibilities that are consistent with limit constraints. Allowance constraints help to preserve this option space, but they are *not* there to further narrow it.

With this representation of design choices, the notion of a design "closing" can be expressed more formally than merely saying that there is some design for which requirements can be met. More careful would be to establish allowance constraints that can be checked for consistency with requirements.

A deliberate decision to narrow constraints around a selected design commitment is a different matter.

In this case, narrowing the option space is an overt choice *expressed in limit constraints imposed by conceptual Views*, not allowance constraints from realizational Views. It's important to keep these two cases separate in one's mind.

A common example of deliberate limitation is in the sub-allocation of resources or performance contributions. Reports of realizable capability are routinely maintained during early development as an architecture is brought to closure, with allowances made for asserted capability. Eventually though, allocations must be imposed — as limit constraints — to lock in these accommodations. Only at that point can analyses be narrowed.

Note finally the different nature of allowances against realizational *compositions*, compared to Elements at the terminal level. Among the possibilities one might want to preserve in a composition are different Relationships, where an Element might be composed in various ways.[47]

In similar fashion, as functional or other Relationships converge, as Trades are settled, and so on, and the need or merit of leaving choices open declines, it is preferable at some point to foreclose other possibilities. Allowances can preserve such choices, but limits lock them in.

Integrated Scenarios

Describe any fully elaborated Scenarios that can be as properly associated with this realizational Element, given the integration of its constituent Elements.

▼ Integrated Scenarios aggregate information relevant to the activities or operation of an integrated system.

An integrated Scenario is associated with some realizational Element, because the full implications of coupling within realizational Elements can only be addressed in this context. In this way, integrated Scenarios clearly illustrate the interconnectedness of an architecture, despite the separation of concerns highlighted in conceptual Views.

The associated realizational Element is generally some composite Element, firstly because the conceptual Scenarios from which they flow are themselves about interactions among Elements, and secondly because realizational coupling extends the breadth of these interactions.

This does not mean that every composite realizational Element will be associated with some integrated Scenario. To the contrary, most integrated Scenarios will be associated with only a few composite realizational Elements — typically those that dominate the overall architectural composition. Nonetheless, one should be prepared to apply this notion wherever it is of use in understanding a system as a system.

Integrated Scenarios may be **technical**, describing operational plans, procedures, and contingencies for the intended use of an architected system, as necessary to carry out its Function; or they may be **programmatic**, describing the integrated schedule for system development, planned integration flow for system assembly, and other processes.

Every integrated Scenario should have a clear purpose and objectives.

The aim is not merely to suggest something that might happen, or to describe the way some activity must happen. Instead, the set of integrated Scenarios collectively establish the cases by which the system architecture is analyzed and validated. Each integrated Scenario should have a defined role within this suite.

Scenarios should also be described by the range of variations they cover, with the objective being to help characterize margins and demonstrate flexibility in the system. Variations can be with respect to parameters, event ordering, optional parts, branch conditions, fault cases, and so on.

An integrated Scenario begins with one or more conceptual Scenarios, which are necessarily narrow in purview (and usually simple in comparison). The Scenario is then elaborated by following to their logical conclusion all the implications of coupling across conceptual issues (with caveats noted below regarding closure criteria), as effected through the integration of realizational design.

Given suitable methods of behavior representation in constraints and conceptual Scenarios, integrated Scenarios should largely be constructible in an automatic manner.

The starting conceptual Scenarios usually encompass some shared circumstance of interest, where one or a few **motivating activities** are accompanied by ancillary Scenarios regarding health, safety, resources, or **other governing concerns**.

Proper association with a particular realizational Element is typically a matter of broadening the scope of elaboration, as coupling pathways are explored, until **closure of some sort** is reached. For example, a spacecraft delta-V Scenario, which at the conceptual level involves a rocket and a mass, multiplies dramatically in scope at the realizational level, as propellant management, pointing, and other coupled behaviors are invoked. Many additional system Elements now participate, which in turn bring into consideration power, thermal, computing, and so on until much of the spacecraft is directly engaged in some manner. The resulting competition for resources then unavoidably affects much of the remainder of the spacecraft. Moreover, because the purpose of the Scenario in the first place is a specified change in velocity, this activity really isn't complete until confirmed, which broadens the scope now to the entire flight-ground navigation loop and all other ancillary functions with coupled interests. In this case then, without further scoping provisions, the realizational Element properly associated with a fully elaborated delta-V Scenario is nearly the entire system!

Given the potential for *every* Scenario to burgeon in like manner, judgment is required to declare appropriate closure criteria for each Scenario. This could be a matter of depth, where certain conceptual matters are set aside (perhaps because added provisions in the Scenario forestall problems in these areas[48]); or it could be a matter of focus, where for narrower interests, a particular realizational Element is targeted for consideration.[49]

It is typically necessary, when addressing any issue of integration, to acknowledge that "integrated" is not a uniquely defined term. Not all Scenarios will apply in all cases, so this is a good reason to be particular about the realizational Element associated with an integrated Scenario.

For example, in the course of its integration and operation, a system usually progresses through a series of deployed configurations. These **deployments** differ by composition, where parts of the system are yet to be integrated, or are present only in temporary emulated form, or have already served their purpose and are jettisoned, disabled, or expended.

Broader deployment differences can encompass external system and environmental Elements, which might be the real thing or only simulated (sometimes at more extreme levels than the real thing).

□

---

[1] A realizational View that mentions a model number, a vendor, a heritage design, or any other established product must always justify this as an architectural choice, traceable to architecturally motivated criteria such as cost or risk. Otherwise, its mention is inappropriate, thereby unnecessarily limiting options or reducing tolerance to variation.

[2] Examples would include multiple serial-numbered units built to a common model number specification. The realizational Element would describe the latter, not the former.

[3] For example, hardware that realizes a reaction wheel for attitude control will typically also realize a mass, a volume, a power load, a temperature zone, a life-limited item, a command sink, a mounted item, a microphonics source, and so on. Depending on circumstances, it may also realize a radiation shield, a safety hazard, or some other conceptual thing.

[4] An example would be separately enumerated RCS thrusters or reaction wheels. These are typically differentiated from one another by location, orientation, and name within the composition, but otherwise must be of the same kind and with uniform properties across the set, such that they are interchangeable. Another example would be in a block redundant set of components, where roles are differentiated only by name and interconnections.

[5] A power concept will define power loads, but will *not* enumerate them. Any Element that realizes a power load becomes a member of the set of power loads. However, each realization of power load is typically a different kind of realizational Element (heater, gyro, receiver, instrument) with varying properties across the set.

[6] Other examples include the set of items that need to be pointed, the set of nodes on a data bus, the set of RCS thrusters, the set of items in a radiation vault, the set of items in a module, and so on, where in each case some conceptual View declares what they must have in common.

[7] Examples would include mechanical assemblies, software modules, celestial bodies, and work breakdown Elements.

[8] Examples include hardware assemblies, software modules, data collections, procedures, and operating plans.

[9] Examples include support systems, services, and facilities.

[10] Examples include ambient fields and celestial bodies.

[11] Programmatic and technical Elements participate in "Subsystem" definitions, products and external system Elements participate in system interface definitions, and so on.

[12] Examples include total mass of a module, and combined performance of some ensemble (as in an error budget).

[13] Examples include placement within a system configuration, enumeration among a set of bus nodes, and assignment of a product to a work breakdown Element.

[14] Examples would include constraints that apply to all power loads, all software modules, all work breakdown Elements, and so on, given that an Element is designated as a realization of such a prototypical item.

[15] For example, a generic constraint for power loads to control their in-rush current to be within trip limits must be supplemented with constraints that establish particular trip limits for each load.

[16] For example, a conceptual View for pointing might identify the need for a star tracker as a particular conceptual Element with specific constraints that can be applied immediately, but would only generically acknowledge items

Continued next page

that contribute to pointing alignments. Once the latter are identified through conceptual mapping, separate specific constraints on each particular alignment Element may then be necessary.

[17] For example, a specific constraint on the downlink performance of a particular transmitter and a particular antenna belongs to neither Element alone. It should be addressed in the View that addresses the realizational composition of transmitter and antenna.

[18] For example, a particular camera with a defined field of view (a volumetric Element) participates in a mutual constraint with all other volumetric Elements of the system that are not fully transparent (the constraint being a null intersection with the field of view). While the constraint involves properties of multiple Elements, it is clearly motivated only by the camera.

[19] Similarly, a particular camera in the architected system could be involved in a pointing constraint that also refers to some external pointing target.

[20] For example, a conceptual View that declares the need for a star tracker, and which then asserts a constraint on its star magnitude sensitivity, is also referring to the specific realizational Element that uniquely realizes that conceptual Element. The same conceptual View may also indicate the need for a set of three reaction wheels; but again, the meaning is three *specific* reaction wheels, for which there is a unique realization.

[21] Examples would include power loads in a conceptual View about power, or alignment Elements in a conceptual View about pointing. In neither case is it likely the prerogative of the View to define what the loads should be or how pointed Elements should be bound to one another for stable alignment.

[22] This occurs when a total mass limit is sub-allocated among the parts of a system.

[23] This occurs when a pointing budget is sub-allocated among contributing pointing error sources.

[24] For example, the same conceptual View about electrical power that identifies the need for a solar array is also the View that assigns specific power constraints to it.

[25] For example, a solar array might be identified as a pointed Element in a conceptual View about electrical power, where a constraint is imposed on its orientation, relative to the Sun. The product that realizes this array thereby becomes one of an enumerated set of pointed Elements in a conceptual View about pointing, but with a pointing constraint specific to the array.

[26] For example, in a conceptual View about system configuration, one might identify all realizational Elements that are non-transparent volumes and are therefore involved in satisfying the field of view constraint for some camera. Similarly, in a conceptual View about pointing, one might identify all realizational Elements that lie along a path through a coordinate frame tree, thereby contributing to end-to-end pointing errors.

[27] For example, these derived constraints might be the individual sub-allocations within a pointing budget.

[28] For example, a mechanical configuration is, in effect, the resolution of many simultaneous constraints on geometric Relationships among the volumetric Elements of the system. A particular configuration establishes the properties of each Relationship (i.e., where each component is relative to the others) such that all mutual constraints on the constituents are met (e.g., free fields of view, proper load paths, limited plume impingement, acceptable end-to-end alignments, etc.), but often without individually constraining the volumetric Elements themselves. Reification of these relationships (typically in the system's mechanical structure) is the means in this case of directing the constraints to a particular component.

[29] For example, the sub-allocation of a mutual constraint between operating mode and power usage divides into an individual mutual constraint on each realizational Element that uses power.

[30] For example, the location of the spacecraft center of mass may be constrained to enable guidance for delta-V, to reduce structural loads, and to limit dynamic coupling.

[31] For example, terminal products in hardware would typically be assemblies considered as deliverable units, though there could be situations where sub-assemblies are architecturally relevant.

[32] These would typically be functionally distinct units. Each should be addressed separately for their particular utilization factors (e.g., different DSN complexes).

[33] A celestial body might be an example, if only its global properties are relevant (e.g., radius or gravitational field).

[34] An internal contamination source is both product and environmental item. A launch vehicle is both external system and environmental item.

[35] For example, consider the separate parts of a control loop (sensors, controllers, actuators, etc.). Each part can be separately realized, but so must the loop, which exists and exhibits required performance only when the parts are integrated.

[36] Examples would include which product is assigned to which work breakdown Element, or which power switch is assigned to which load.

Continued next page

[37] Examples would include a particular chosen arrangement of system mechanical configuration items.
[38] Electronic cards sharing a chassis comprise a composition with interdependencies that depend on relative placement (e.g., thermal, shielding, interference…). Particular placement would not be specified in any concept, but must be defined in realization of the composition in order to address issues that *are* specified in concepts.
[39] For example, one might be interested in the inertial properties of a platform, or the latest delivery date of all products needed for some test deployment.
[40] For example, multiple data sources may need to compete for bandwidth on the same bus, and if so, they will need to cooperate in its utilization.
[41] For example, one would expect to see a repeated pattern for composing hardware components pairwise with assigned management functions in software. Each would be a specialization of this pattern, but all would be integrated in effectively the same manner.
[42] For example, a conceptual View for data interactions might reasonably define the need for a common bus adapter design across all units attached to a data bus. It would *not* be reasonable to map a conceptual bus adapter to the entire realizational unit within which a real bus adapter appears, especially if other Relationships (e.g., delivery responsibility) apply uniquely to the part and not the whole.
[43] Examples would include a dependence of minimum mass on the level of needed functional performance, or the coupling of power usage to operating mode.
[44] For example, an implementation that relies on radioactive material (e.g., as a calibration source) would not be fully described without properties quantifying this feature. However, in an architecture with no concept for how to handle self-produced nuclear radiation, these properties could be disruptive.
[45] Examples include the addition of idle/off states, transitional states, and/or failure modes to the set of states that were conceptually defined as essential behavior.
[46] Examples would include convergence on a system configuration or a mission design, selection among a vendor's standard offerings, making a technology choice, or assignment of delivery responsibility.
[47] For example, there may be technical reasons to leave options open for the location of a hardware Element (e.g., different modules, chassis, busses, etc.). Programmatic Relationships may also need to remain open, including Relationships between programmatic and technical Element (such as which work breakdown Element is responsible for a particular product).
[48] For example, with asserted initial conditions that mitigate energy concerns, a Scenario might be safely truncated with regard to its power implications.
[49] For example, if one's interest in specifically in flight system capabilities, a delta-V Scenario might be limited to the flight system Element.

■

Appendix C View Contributions to Gate Products

Table 1: Products and Views required at MCR

Gate Product	WBS Element	Required Maturity	Contributing Views and Framework Elements
Project Formulation Authorization Document or equivalent	01.01	Ready to sign	*N/A - not intended to be part of the AD*
L1 requirements	01.01	Prelim	Stakeholders Concerns (Success criteria)
Partnerships and interagency and international agreements	01.01	Prelim	*N/A - not intended to be part of the AD*
Terms of Reference (for Life Cycle Reviews)	01.01	Draft	*N/A - not intended to be part of the AD*
Project Task Plan	01.01	Phase A	Work Breakdown
Formulation Agreement	01.01	Phase A - Baseline Phase B - Prelim	Project Lifecycle Project Planning
Business Plan	01.02	Approach	Project Planning
Launch Approval Engineering Plan	01.06	ECLASS	*N/A - not intended to be part of the AD*
Project Review Plan	01.01	Prelim	Project Lifecycle
Project Acquisition Plan	01.01	Prelim	*N/A - not intended to be part of the AD*
Export Compliance Management Plan	01.01	Initial	*N/A - not intended to be part of the AD*
SEMP	02.01	Prelim	Systems Engineering
Risk Management Plan	02.11	Approach	Risk Management
Tech Development Plan	02.01	Baseline	Systems Engineering
Project V&V Plan	02.10	Approach	Verification and Validation
Project WBS & Dictionary	01.02	Prelim	Work Breakdown
Project integrated life-cycle network schedules	01.02	Prelim	*N/A - not intended to be part of the AD*
Cost Estimates	01.02	Model	*N/A - not intended to be part of the AD*
Life-cycle budget	01.02	Proposed	*N/A - not intended to be part of the AD*
Life-cycle workforce plan	01.02	Proposed	*N/A - not intended to be part of the AD*
Work Agreements/Summary Work Agreements	01.02	Phase A	*N/A - not intended to be part of the AD*
Infrastructure requirements and plans, business case analysis for infrastructure (if applicable)	01.02	Initial	*N/A - not intended to be part of the AD*
Project L2 Requirements	02.01	Prelim	Project System realizational View (list of RQ)
Mission Architecture	02.01	Prelim	Project System realizational View various Conceptual Views
Mission Concept & Operating Scenarios	02.01	Prelim	Integrated Scenarios from Project System realizational View
Initial technology/engineering dev/heritage assessments	02.01	Initial	Technology Development Analysis
Significant Risk List	02.11	Initial	*N/A - not intended to be part of the AD*
Science L2 Requirements	04.01	Prelim	Science Dataset realizational View Science System realizational View
Payload L3 Requirements	05.02	Prelim	Payload realizational View (list of RQ)
Payload Architecture	05.02	Prelim	Payload realizational View (Elements, interfaces and Functions)
Spacecraft L3 Requirements	06.02	Prelim	Spacecraft realizational View (list of RQ)
Spacecraft Architecture	06.02	Prelim	Spacecraft realizational View (Elements, interfaces and Functions)
Mission Ops Concept document	07.02	Initial	Operations Concept conceptual View
MOS L3 Requirements	07.02	Key	MOS realizational View
Ground Architecture	07.02	Concept	MOS realizational View Science System realizational View
FS I&T Plan	10.02	Approach	FS I&T realizational View (Elements, Scenarios)

Table 2: Gate Products and Views required at SRR

Gate Product	WBS Element	Required Maturity	Contributing Views and Framework Elements
L1 requirements	1.01	Baseline	Stakeholders Concerns (Success criteria)
Acquisition Strategy Meeting (ASM) minutes	01.01	Final	N/A - not intended to be part of the AD
Terms of Reference (for Life Cycle Reviews)	01.01	Baseline	N/A - not intended to be part of the AD
Business Plan	01.02	Prelim	Project Planning
Project Review Plan	01.01	Baseline	Project Lifecycle
Project Acquisition Plan	01.01	Baseline	N/A - not intended to be part of the AD
SEMP	02.01	Baseline	Systems Engineering
Project SW Management Plan	02.02	Prelim	Software Development
Information & Configuration Mgmt. (ICM) Plan	02.04	Baseline	Configuration Management
Risk Management Plan	02.11	Baseline	Risk Management
Tech Development Plan	02.01	Update	Systems Engineering
Mission Assurance Plan	03.01	Initial	Mission Assurance
Project L2 Requirements	02.01	Baseline	Project System realizational View (list of RQ)
Mission Architecture	02.01	Baseline	Project System realizational View various Conceptual Views
Initial technology/engineering dev/heritage assessments	02.01	Update	Technology Development Analysis
IT Security Plan	02.02	Prelim	N/A - not intended to be part of the AD
Environmental Requirements Document	03.03	Prelim	Environmental Compatibility
Significant Risk List	02.11	Update	N/A - not intended to be part of the AD
Science L2 Requirements	04.01	Baseline	Science Dataset realizational View Science System realizational View
Payload L3 Requirements	05.02	Baseline	Payload realizational View (list of RQ)
L4 Instrument Requirements	05.xx	Key/ Driving	Instrument realizational Views (list of RQ)
Spacecraft L3 Requirements	06.02	Baseline	Spacecraft realizational View (list of RQ)
Spacecraft Architecture	06.02	Baseline	Spacecraft realizational View (Elements, interfaces and Functions)
L4 Spacecraft Subsystem Requirements	06.xx	Key/ Driving	Subsystem realizational Views
Mission Ops Concept document	07.02	Update	Operations Concept conceptual View
MOS L3 Requirements	07.02	Baseline	MOS realizational View
Ground Architecture	07.02	Prelim	MOS realizational View Science System realizational View
Launch Services Requirements Document	02.09	Prelim	Launch System realizational View

Table 3: Gate Products and Views required at MDR

Gate Product	WBS Element	Required Maturity	Contributing Views and Framework Elements
L1 requirements	1.01	Update	Stakeholders Concerns (Success criteria)
Partnerships and interagency and international agree-ments	01.01	Baseline	N/A - not intended to be part of the AD
Software Independent Verification and Validation Plan (for projects with IV&V)	03.08	Prelim	N/A - not intended to be part of the AD
Acquisition Strategy Meeting (ASM) minutes	01.01	Final	N/A - not intended to be part of the AD
Planetary Protection Certification	02.06	Final	N/A - not intended to be part of the AD
Terms of Reference (for Life Cycle Reviews)	01.01	Baseline	N/A - not intended to be part of the AD

Gate Product	WBS Element	Required Maturity	Contributing Views and Framework Elements
Project Task Plan	01.01	Phase B	Work Breakdown
Formulation Agreement	01.01	Phase B - Baseline	Project Lifecycle Project Planning
Project Plan	01.01	Prelim	Project Planning
PIP 1.2 - Project Manager's Decisions, Guidance and Policies	01.01	Baseline	N/A - not intended to be part of the AD
Business Plan	01.02	Baseline	Project Planning
Launch Approval Engineering Plan	01.06	Baseline	N/A - not intended to be part of the AD
PIP 1.5 - Risk Communication Plan	01.06	Baseline	N/A - not intended to be part of the AD
Project Review Plan	01.01	Update	Project Lifecycle
Project Acquisition Plan	01.01	Update	N/A - not intended to be part of the AD
Export Compliance Management Plan	01.01	Baseline	N/A - not intended to be part of the AD
SEMP	02.01	Baseline	Systems Engineering
Project SW Management Plan	02.02	Baseline	Software Development
Information & Configuration Mgmt. (ICM) Plan	02.04	Baseline	Configuration Management
Risk Management Plan	02.11	Baseline	Risk Management
Tech Development Plan	02.01	Baseline	Systems Engineering
Project V&V Plan	02.10	Prelim	Verification and Validation
System Safety Plan	03.02	Prelim	Safety
Mission Assurance Plan	03.01	Prelim	Mission Assurance
Education Plan	11.01	Prelim	N/A - not intended to be part of the AD
Communications Plan	11.01	Prelim	N/A - not intended to be part of the AD
Mission Projects Design, Verification/Validation and Operations Principles Compliance Matrix	02.01	Prelim	Design Principles Compliance Analysis
Mission Projects Flight Project Practices Compliance Matrix	01.01	Prelim	Flight Project Practice Compliance Analysis
Project WBS & Dictionary	01.02	Prelim	Work Breakdown
Project integrated life-cycle network schedules	01.02	Prelim	N/A - not intended to be part of the AD
Cost Estimates	01.02	Prelim	N/A - not intended to be part of the AD
Life-cycle budget	01.02	Prelim	N/A - not intended to be part of the AD
Life-cycle workforce plan	01.02	Prelim	N/A - not intended to be part of the AD
Work Agreements/Summary Work Agreements	01.02	Phase B	N/A - not intended to be part of the AD
Project Space Asset Protection Plan	01.01	Prelim	Space Asset Protection
Project L2 Requirements	02.01	Baseline	Project System realizational View (list of RQ)
Mission Architecture	02.01	Baseline	Project System realizational View various Conceptual Views
Mission Concept & Operating Scenarios	02.01	Baseline	Integrated Scenarios from Project System realizational View
Initial technology/engineering dev/heritage assessments	02.01	Update	Technology Development Analysis
IT Security Plan	02.02	Baseline	N/A - not intended to be part of the AD
Environmental Requirements Document	03.03	Prelim	Environmental Compatibility
Significant Risk List	02.11	Prelim	N/A - not intended to be part of the AD
Design Report	02.01	Prelim	Design Compliance Analysis
Science L2 Requirements	04.01	Baseline	Science Dataset realizational View Science System realizational View
Payload L3 Requirements	05.02	Baseline	Payload realizational View (list of RQ)
Payload Architecture	05.02	Baseline	Payload realizational View (Elements, interfaces and Functions)
Payload/Instrument Design	05.02	Initial	N/A - not intended to be part of the AD
L4 Instrument Requirements	05.xx	Key/ Driving	Instrument realizational Views (list of RQ)
Spacecraft Operating Scenarios	06.02	Key/ Driving	Integrated Scenarios from Spacecraft realizational View
Spacecraft L3 Requirements	06.02	Baseline	Spacecraft realizational View (list of RQ)
Spacecraft Architecture	06.02	Baseline	Spacecraft realizational View (Elements, interfaces and Functions)

Gate Product	WBS Element	Required Maturity	Contributing Views and Framework Elements
Spacecraft System Design	06.02	Initial	N/A - not intended to be part of the AD
Spacecraft Subsystem (S/S) Design	06.xx	Initial	N/A - not intended to be part of the AD
L4 Spacecraft Subsystem Requirements	06.xx	Key/ Driving	Subsystem realizational Views
Mission Ops Concept document	07.02	Prelim	Operations Concept conceptual View
MOS L3 Requirements	07.02	Baseline	MOS realizational View
Ground Data System (GDS) Requirements Document	07.02	Prelim	GDS realizational View (list of RQ)
Ground Architecture	07.02	Baseline	MOS realizational View Science System realizational View
MOS Functional Design Document	07.02	Approach	MOS realizational View
Ground Station/Multi-Mission Service Provider Agreements	07.02	Draft	N/A - not intended to be part of the AD
MOS Verification & Validation Plan	07.14	Approach	Verification and Validation
Launch Services Requirements Document	02.09	Baseline	Launch System realizational View
FS I&T Plan	10.02	Prelim	FS I&T realizational View (Elements, Scenarios)

Table 4: Gate Products and Views required at PDR

Gate Product	WBS Element	Required Maturity	Contributing Views and Framework Elements
NASA NEPA compliance documentation	01.01	Assessed	N/A - not intended to be part of the AD
Partnerships and interagency and international agreements	01.01	Baseline	N/A - not intended to be part of the AD
Software Independent Verification and Validation Plan (for projects with IV&V)	03.08	Baseline	N/A - not intended to be part of the AD
Launch System ICD	02.01	Prelim	N/A - not intended to be part of the AD
Range Safety Risk Management Plan	03.02	Prelim	N/A - not intended to be part of the AD
Project Task Plan	01.01	Phase CD	Work Breakdown
Project Plan	01.01	Baseline	Project Planning
Business Plan	01.02	Update	Project Planning
Mishap Preparedness and Contingency Plan	01.01	Baseline	N/A - not intended to be part of the AD
Project SW Management Plan	02.02	Update	Software Development
Information & Configuration Mgmt. (ICM) Plan	02.04	Update	Configuration Management
Planetary Protection Plan	02.06	Baseline	Planetary Protection
Contamination Control Plan	02.07	Baseline	Contamination Control
Risk Management Plan	02.11	Update	Risk Management
Tech Development Plan	02.01	Update	Systems Engineering
Project V&V Plan	02.10	Baseline	Verification and Validation
System Safety Plan	03.02	Baseline	Safety
Mission Assurance Plan	03.01	Baseline	Mission Assurance
PIP 3.3 - Reliability Assurance Plan	03.04	Baseline	Mission Assurance
PIP 3.4 - Electronic Parts Program Plan	03.05	Baseline	Mission Assurance
PIP 3.5 - Quality Assurance Plan	03.06 03.07	Baseline	Mission Assurance
PIP 3.6 - Problem Reporting Plan	03.04	Baseline	Mission Assurance
PIP 4.1 - Science Management Plan	04.01	Baseline	N/A - not intended to be part of the AD
PIP4.2 - Science Data Management and Archive Plan	04.01	Prelim	Science Data Management conceptual View
PIP 5.1 - Payload System Implementation Plan (Note 15)	05.01	Baseline	Payload Engineering
PIP 6.1 - Spacecraft System Implementation Plan (Note 15)	06.01	Baseline	Spacecraft Engineering
PIP 6.2 - Materials and Processes Plan	06.13	Baseline	N/A - not intended to be part of the AD
PIP 7.1 - Mission Operations System Implementation Plan (including GDS) (Note 15)	07.01	Baseline	Mission System Engineering

Gate Product	WBS Element	Required Maturity	Contributing Views and Framework Elements
PIP 8.1 - Launch Services Implementation Plan (Note 15)	02.09	Baseline	N/A - not intended to be part of the AD
Education Plan	11.01	Baseline	N/A - not intended to be part of the AD
Communications Plan	11.01	Baseline	N/A - not intended to be part of the AD
Mission Projects Design, Verification/Validation and Operations Principles Compliance Matrix	02.01	Baseline	Design Principles Compliance Analysis
Mission Projects Flight Project Practices Compliance Matrix	01.01	Baseline	Flight Project Practice Compliance Analysis
Project WBS & Dictionary	01.02	Final	Work Breakdown
Project integrated life-cycle network schedules	01.02	Baseline	N/A - not intended to be part of the AD
Cost Estimates	01.02	Baseline	N/A - not intended to be part of the AD
Life-cycle budget	01.02	Baseline	N/A - not intended to be part of the AD
Life-cycle workforce plan	01.02	Baseline	N/A - not intended to be part of the AD
Work Agreements/Summary Work Agreements	01.02	Phase CD	N/A - not intended to be part of the AD
Project Space Asset Protection Plan	01.01	Baseline	Space Asset Protection
Project L2 Requirements	02.01	Update	Project System realizational View (list of RQ)
Mission Architecture	02.01	Update	Project System realizational View various Conceptual Views
Mission Concept & Operating Scenarios	02.01	Update	Integrated Scenarios from Project System realizational View
IT Security Plan	02.02	Update	N/A - not intended to be part of the AD
Environmental Requirements Document	03.03	Baseline	Environmental Compatibility
Significant Risk List	02.11	Baseline	N/A - not intended to be part of the AD
Design Report	02.01	Prelim	Design Compliance Analysis
Inter-system (flight-ground) interfaces (DRAFT ICDs)	02.01	Draft	N/A - not intended to be part of the AD
Probabilistic Risk Assessment (when applicable)	03.04	Initial	N/A - not intended to be part of the AD
Functional FMECA (Risk Class A projects only)	03.04	Prelim	N/A - not intended to be part of the AD
Telecommunications Design Control Document	06.06	Prelim	N/A - not intended to be part of the AD
Mission/System Fault Tree	02.01	Prelim	N/A - not intended to be part of the AD
Test-As-You-Fly Exceptions	02.10	Prelim	N/A - not intended to be part of the AD
Lessons Learned - Formulation	02.01	Final	N/A - not intended to be part of the AD
MOU with Science Data Archive	04.01	Baseline	N/A - not intended to be part of the AD
Payload L3 Requirements	05.02	Update	Payload realizational View (list of RQ)
Payload Architecture	05.02	Update	Payload realizational View (Elements, interfaces and Functions)
Payload/Instrument Design	05.02	Prelim	N/A - not intended to be part of the AD
L4 Instrument Requirements	05.xx	Baseline	Instrument realizational Views (list of RQ)
Instrument Calibration Requirements and Plan	05.xx	Prelim	Instrument realizational Views (list of RQ, Scenarios)
Spacecraft Operating Scenarios	06.02	Baseline	Integrated Scenarios from Spacecraft realizational View
Spacecraft L3 Requirements	06.02	Update	Spacecraft realizational View (list of RQ)
Spacecraft Architecture	06.02	Update	Spacecraft realizational View (Elements, interfaces and Functions)
Spacecraft System Design	06.02	Prelim	N/A - not intended to be part of the AD
Spacecraft -Instrument interfaces (ICDs)	06.02	Draft	N/A - not intended to be part of the AD
Spacecraft subsystem interfaces (ICDs)	06.02	Draft	N/A - not intended to be part of the AD
Spacecraft System Software Mgmt Plans	06.02	Baseline	N/A - not intended to be part of the AD
S/C System Software Requirements Document	06.02	Baseline	Flight Software Realizational View
Spacecraft System Software Architectural Design	06.02	Baseline	Flight Software Realizational View
Spacecraft Subsystem (S/S) Design	06.xx	Prelim	N/A - not intended to be part of the AD
L4 Spacecraft Subsystem Requirements	06.xx	Baseline	Subsystem realizational Views
Mission Ops Concept document	07.02	Baseline	Operations Concept conceptual View
MOS L3 Requirements	07.02	Update	MOS realizational View
Ground Data System (GDS) Requirements Document	07.02	Baseline	GDS realizational View (list of RQ)
Ground Architecture	07.02	Update	MOS realizational View Science System realizational View

Gate Product	WBS Element	Required Maturity	Contributing Views and Framework Elements
GDS Software Management Plan	09.02	Baseline	N/A - not intended to be part of the AD
MOS Functional Design Document	07.02	Prelim	MOS realizational View
Ground Station/Multi-Mission Service Provider Agreements	07.02	Prelim	N/A - not intended to be part of the AD
MOS Verification & Validation Plan	07.14	Prelim	Verification and Validation
MOS Training Plan	07.02	Approach	N/A - not intended to be part of the AD
GDS Integration & Test (I&T) Plan	09.17	Prelim	N/A - not intended to be part of the AD
Software Interface Specifications (SISs)	09.02	List	N/A - not intended to be part of the AD
Launch Services Requirements Document	02.09	Baseline	Launch System realizational View
FS I&T Plan	10.02	Baseline	FS I&T realizational View (Elements, Scenarios)
Mission Plan, Including Mission Scenarios	12.03	Prelim	Project System realizational View (list of Scenarios)
Navigation Plan	12.04	Prelim	Navigation conceptual View MOS realizational View
Target Specification Document	12.02	Prelim	N/A - not intended to be part of the AD

Table 5: Gate Products and Views required at CDR

Gate Product	WBS Element	Required Maturity	Contributing Views and Framework Elements
Launch System ICD	02.01	Baseline	N/A - not intended to be part of the AD
Project Task Plan	01.01	Phase CD	Work Breakdown
Project V&V Plan	02.10	Update	Verification and Validation
Mission Assurance Plan	03.01	Update	Mission Assurance
PIP4.2 - Science Data Management and Archive Plan	04.01	Baseline	Science Data Management conceptual View
Education Plan	11.01	Update	N/A - not intended to be part of the AD
Communications Plan	11.01	Update	N/A - not intended to be part of the AD
Project integrated life-cycle network schedules	01.02	Update	N/A - not intended to be part of the AD
Cost Estimates	01.02	Update	N/A - not intended to be part of the AD
Life-cycle workforce plan	01.02	Update	N/A - not intended to be part of the AD
Project Space Asset Protection Plan	01.01	Update	Space Asset Protection
Environmental Requirements Document	03.03	Update	Environmental Compatibility
Significant Risk List	02.11	Update	N/A - not intended to be part of the AD
Design Report	02.01	Baseline	Design Compliance Analysis
Inter-system (flight-ground) interfaces (DRAFT ICDs)	02.01	Baseline	N/A - not intended to be part of the AD
Probabilistic Risk Assessment (when applicable)	03.04	Update	N/A - not intended to be part of the AD
Functional FMECA (Risk Class A projects only)	03.04	Baseline	N/A - not intended to be part of the AD
Telecommunications Design Control Document	06.06	Baseline	N/A - not intended to be part of the AD
Incompressible Test List	02.10	Prelim	N/A - not intended to be part of the AD
Mission/System Fault Tree	02.01	Baseline	N/A - not intended to be part of the AD
Test-As-You-Fly Exceptions	02.10	Baseline	N/A - not intended to be part of the AD
Payload/Instrument Design	05.02	Baseline	N/A - not intended to be part of the AD
Instrument Calibration Requirements and Plan	05.xx	Baseline	Instrument realizational Views (list of RQ, Scenarios)
Spacecraft Operating Scenarios	06.02	Update	Integrated Scenarios from Spacecraft realizational View
Spacecraft System Design	06.02	Baseline	N/A - not intended to be part of the AD
Spacecraft -Instrument interfaces (ICDs)	06.02	Baseline	N/A - not intended to be part of the AD
Spacecraft subsystem interfaces (ICDs)	06.02	Baseline	N/A - not intended to be part of the AD
Spacecraft design verification requirements matrix	06.02	Prelim	N/A - not intended to be part of the AD
Flight Sequences (launch, and mission critical)	06.02	Prelim	N/A - not intended to be part of the AD
System Testbed Integration & Test Plan	06.13	Baseline	N/A - not intended to be part of the AD

Gate Product	WBS Element	Required Maturity	Contributing Views and Framework Elements
Command and telemetry dictionaries	06.02	Prelim	N/A - not intended to be part of the AD
Spacecraft System Software Detailed Design	06.02	Baseline	N/A - not intended to be part of the AD
Flight Rules and Constraints Document	06.02	Prelim	N/A - not intended to be part of the AD
Spacecraft Subsystem (S/S) Design	06.xx	Baseline	N/A - not intended to be part of the AD
Ground Data System (GDS) Requirements Document	07.02	Update	GDS realizational View (list of RQ)
MOS Functional Design Document	07.02	Baseline	MOS realizational View
Ground Station/Multi-Mission Service Provider Agreements	07.02	Baseline	N/A - not intended to be part of the AD
MOS Verification & Validation Plan	07.14	Baseline	Verification and Validation
MOS Training Plan	07.02	Prelim	N/A - not intended to be part of the AD
GDS Integration & Test (I&T) Plan	09.17	Baseline	N/A - not intended to be part of the AD
Operations Interface Agreements (OIAs)	07.02	List	MOS realizational View (list of team-to-team interfaces)
Software Interface Specifications (SISs)	09.02	Baseline	N/A - not intended to be part of the AD
Flight Operations Plan	07.02	Prelim	N/A - not intended to be part of the AD
Operations Procedures	07.02	List	N/A - not intended to be part of the AD
MOS Contingency Plans and Procedures	07.02	List	N/A - not intended to be part of the AD
Flight Rules and Constraints Check Matrix	07.02	Prelim	N/A - not intended to be part of the AD
Sequence Component Dictionary	07.02	Prelim	N/A - not intended to be part of the AD
FS I&T Plan	10.02	Update	FS I&T realizational View (Elements, Scenarios)
Mission Plan, Including Mission Scenarios	12.03	Baseline	Project System realizational View (list of Scenarios)
Navigation Plan	12.04	Baseline	Navigation conceptual View MOS realizational View

Table 6: Gate Products and Views required at SIR

Gate Product	WBS Element	Required Maturity	Contributing Views and Framework Elements
Range Safety Risk Management Plan	03.02	Baseline	N/A - not intended to be part of the AD
PIP4.2 - Science Data Management and Archive Plan	04.01	Update	Science Data Management conceptual View
Work Agreements/Summary Work Agreements	01.02	Update	N/A - not intended to be part of the AD
Design Report	02.01	Update	Design Compliance Analysis
Incompressible Test List	02.10	Baseline	N/A - not intended to be part of the AD
Test-As-You-Fly Exceptions	02.10	Update	N/A - not intended to be part of the AD
Mission Operations Assurance Plan	03.09	Prelim	N/A - not intended to be part of the AD
Payload/Instrument Design	05.02	Update	N/A - not intended to be part of the AD
Spacecraft System Design	06.02	Update	N/A - not intended to be part of the AD
Spacecraft design verification requirements matrix	06.02	Baseline	N/A - not intended to be part of the AD
Command and telemetry dictionaries	06.02	Baseline	N/A - not intended to be part of the AD
Spacecraft System Software Detailed Design	06.02	Baseline	N/A - not intended to be part of the AD
Flight Rules and Constraints Document	06.02	Baseline	N/A - not intended to be part of the AD
Spacecraft Subsystem (S/S) Design	06.xx	Update	N/A - not intended to be part of the AD
MOS Functional Design Document	07.02	Update	MOS realizational View
MOS Training Plan	07.02	Baseline	N/A - not intended to be part of the AD
GDS Integration & Test (I&T) Plan	09.17	Update	N/A - not intended to be part of the AD
Operations Interface Agreements (OIAs)	07.02	Baseline	N/A - not intended to be part of the AD
Sequences for ATLO testing and Launch/Early Flight Operations	07.02	Baseline	N/A - not intended to be part of the AD
Flight Operations Plan	07.02	Baseline	N/A - not intended to be part of the AD
Operations Procedures	07.02	Baseline	N/A - not intended to be part of the AD
MOS Contingency Plans and Procedures	07.02	Baseline	N/A - not intended to be part of the AD

Gate Product	WBS Element	Required Maturity	Contributing Views and Framework Elements
Flight Rules and Constraints Check Matrix	07.02	Baseline	N/A - not intended to be part of the AD
Sequence Component Dictionary	07.02	Baseline	N/A - not intended to be part of the AD
Launch Site Responsibility Matrix	02.09	Prelim	N/A - not intended to be part of the AD
FS I&T Plan	10.02	Update	FS I&T realizational View (Elements, Scenarios)
Target Specification Document	12.04	Baseline	N/A - not intended to be part of the AD

Table 7: Gate Products and Views required at PSR

Gate Product	WBS Element	Required Maturity	Contributing Views and Framework Elements
Launch System ICD	02.01	Update	N/A - not intended to be part of the AD
Certificate of Flight Readiness	02.01	Final	various Realizational Views various Analyses
As-Built Documents	05.xx	Baseline	N/A - not intended to be part of the AD
Flight Sequences (launch, and mission critical)	06.02	Baseline	N/A - not intended to be part of the AD
Testbed Validation Report	06.14	Baseline	N/A - not intended to be part of the AD
Launch Site Responsibility Matrix	02.09	Baseline	N/A - not intended to be part of the AD
Spacecraft design verification results	06.13	Baseline	N/A - not intended to be part of the AD
Spacecraft Idiosyncrasies Document	06.13	Prelim	N/A - not intended to be part of the AD
Transportation Plan	06.12	Final	N/A - not intended to be part of the AD

Table 8: Gate Products and Views required at ORR

Gate Product	WBS Element	Required Maturity	Contributing Views and Framework Elements
Project Task Plan	01.01	Phase E	Work Breakdown
Work Agreements/Summary Work Agreements	01.02	Phase E	N/A - not intended to be part of the AD
Decommissioning/Disposal Plan	01.01	Baseline	Project System realizational View (Decom-missioning Scenario)
Launch - Hold Criteria	02.01	Prelim	N/A - not intended to be part of the AD
Project Verification & Validation Results	02.10	Prelim	N/A - not intended to be part of the AD
Mission Operations Assurance Plan	03.09	Baseline	N/A - not intended to be part of the AD
Instrument Calibration Data	05.xx	Baseline	N/A - not intended to be part of the AD
Command and telemetry dictionaries	06.02	Update	N/A - not intended to be part of the AD
Flight Rules and Constraints Document	06.02	Update	N/A - not intended to be part of the AD
Sequences for ATLO testing and Launch/Early Flight Operations	07.02	Final	N/A - not intended to be part of the AD
Flight Operations Plan	07.02	Update	N/A - not intended to be part of the AD
Sequence Component Dictionary	07.02	Baseline	N/A - not intended to be part of the AD
Mission Sequence Plan	07.02	Baseline	N/A - not intended to be part of the AD
Mission Plan, Including Mission Scenarios	12.03	Update	Project System realizational View (list of Scenarios)
Navigation Plan	12.04	Update	Navigation conceptual View MOS realizational View
Target Specification Document	12.02	Update	N/A - not intended to be part of the AD

Table 9: Gate Products and Views required at MRR

Gate Product	WBS Element	Required Maturity	Contributing Views and Framework Elements
Probabilistic Risk Assessment (when applicable)	03.04	Update	N/A - not intended to be part of the AD
Launch - Hold Criteria	02.01	Baseline	N/A - not intended to be part of the AD
Project Verification & Validation Results	02.10	Baseline	N/A - not intended to be part of the AD
End of Mission Plan	02.01	Baseline	N/A - not intended to be part of the AD
Testbed Validation Report	06.14	Update	N/A - not intended to be part of the AD

■

Appendix D Bibliography

[1] R. Rasmussen, J. Day and S. Jenkins, "System Architecting Methodology Task Report," April 2020. [Online]. Available: https://wiki.jpl.nasa.gov/download/attachments/264051817/System%20Architecting%20Methodology%20Task%20Report.pdf?api=v2.

[2] J. J. Odell, "Six Different Kinds of Composition," in *Advanced Object-Oriented Analysis and Design Using UML*, Cambridge University Press, 1998.

[3] DoD, "Systems Engineering Fundamentals," January 2001. [Online]. Available: https://apps.dtic.mil/docs/citations/ADA606327.

[4] IEEE Std 1471-2000, "Recommended Practice for Architectural Description of Software-Intensive Systems," [Online]. Available: https://bravo-lib.jpl.nasa.gov/docushare/dsweb/Get/Document-119114/IEEE-STD-1471-2000.pdf.

[5] Kruchten, "4+1 View Model," [Online]. Available: http://www.softwarearchitectures.com/library/resources/Kruchten4+1.pdf.

[6] OMG, "Model Driven Architecture," [Online]. Available: http://www.omg.org/mda/.

[7] ISO/IEC 10746, "The Reference Model for Open Distributed Processing (RM-ODP)," [Online]. Available: http://www.rm-odp.net/.

[8] CCSDS, "Reference Architecture for Space Data Systems (RASDS)," [Online]. Available: https://public.ccsds.org/Pubs/311x0m1.pdf.

[9] The Open Group, "Architecture Framework (TOGAF)," [Online]. Available: http://www.opengroup.org/togaf/.

[10] ISO/IEC 15288, "Systems and software engineering — System life cycle processes," 15 May 2015. [Online]. Available: https://www.iso.org/standard/63711.html.

[11] NASA SP-2016-6105, "Systems Engineering Handbook," [Online]. Available: https://www.nasa.gov/sites/default/files/atoms/files/nasa_systems_engineering_handbook_0.pdf.

[12] JPL, "Flight Project Practices," [Online]. Available: http://rules.jpl.nasa.gov/pdf/FPP__12.pdf.

[13] NASA/SP-2014-3705, *NASA Space Flight Program and Project Management Handbook,* September 2014.

[14] NASA, *FY2021 Budget Request Executive Summary,* 2020.

[15] Jet Propulsion Laboratory, *Institutional Project Review Plan, DocID 75512 (Rev. 9),* 2019.

[16] B. S. Blanchard and W. J. Fabrycky, Systems Engineering and Analysis, Prentice-Hall Inc., 1981.

[17] Jet Propulsion Laboratory, *JPL Standard Flight Project Work Breakdown Structure Template, DocID 59533 (Rev. 7),* 2018.

[18] Jet Propulsion Laboratory, *Work Breakdown Structure Tailoring, DocID 59535 (Rev. 3),* 2017.

[19] JPL, "Flight Hardware Hierarchy and Nomenclature," [Online]. Available: https://rules.jpl.nasa.gov/cgi/doc-gw.pl?docid=78364.

[20] E. W. Dijkstra, "On the role of scientific thought" in Selected writings on Computing: A Personal Perspective, New York, NY: Springer-Verlag, 1982, p. 60–66.

[21] R. D. Rasmussen, *Architecture Framework Definition, JPL D-55628 (Version 15),* Jet Propulsion Laboratory, 2019.

■