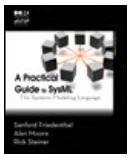

A Practical Guide to SysML: The Systems Modeling Language

by Sanford Friedenthal, Alan Moore and Rick Steiner

Chapter 2: Model-Based Systems Engineering

Overview

Model-based systems engineering (MBSE) applies systems modeling as part of the systems engineering process described in Chapter 1 to support analysis, specification, design, and verification of the system being developed. A primary artifact of MBSE is a coherent model of the system being developed. This approach enhances communications, specification and design precision, design integration, and reuse of system specification and design artifacts.

This chapter summarizes MBSE concepts to provide further context for SysML without emphasizing the specific modeling language, method, or tools. MBSE is contrasted with the more traditional document-based approach to motivate the use of MBSE and highlight its benefits. Principles for effective modeling are also discussed.

2.1 Contrasting the Document-Based and Model-Based Approach

The following sections contrast the document-based approach and the model-based approach to systems engineering.

2.1.1 Document-Based Systems Engineering Approach

Traditionally, large projects have employed a **document-based systems engineering** approach. This approach is characterized by the generation of textual specifications and design documents, in hard-copy or electronic file format, that are then exchanged between customers, users, developers, and testers. System requirements and design information are expressed in these documents and drawings. The systems engineering emphasis is placed on controlling the documentation and ensuring the documents and drawings are valid, complete, and consistent, and that the developed system complies with the documentation.

In the document-based approach, specifications for a particular system, its subsystems, and its hardware and software components are usually depicted in a hierarchical tree, called a **specification tree**. A systems engineering management plan (SEMP) documents how the systems engineering process is employed on the project, and how the engineering disciplines work together to develop the documentation needed to satisfy the requirements in the specification tree. Systems engineering activities are planned by estimating the time and effort required to generate documentation, and progress is then measured by the state of completion of the documents.

Document-based systems engineering typically relies on a concept of operation document to define how the system is used to support the required mission or objective. Functional analysis is performed to allocate the top-level system functions to the components of the system. Drawing tools are used to capture the system design, such as the functional flow diagrams or schematic block diagrams. These diagrams are stored as separate files and included in the system design documentation. Engineering trade studies and analyses are performed and documented by many different disciplines to evaluate and optimize alternative designs and allocate performance requirements. The analysis may be supported by individual analysis models for performance, reliability, safety, mass properties, and other aspects of the system.

Requirements traceability is established and maintained in the document-based approach by tracing requirements between the specifications at different levels of the specification hierarchy. Requirements management tools are used to parse requirements contained in the specification documents and capture them in a requirements database. The traceability between requirements and design is maintained by identifying the part of the system or subsystem that satisfies the requirement, and/or the verification procedures used to verify the requirement, and then reflecting this in the requirements database.

The document-based approach can be rigorous but has some fundamental limitations. The completeness, consistency, and relationships between requirements, design, engineering analysis, and test information are difficult to assess since this information is spread across several documents. This makes it difficult to understand a particular aspect of the system and to perform the necessary traceability and change impact assessments. This, in turn, leads to poor synchronization between system-level requirements and design and lower-level hardware and software design. It also makes it difficult to maintain or reuse the system requirements and design information for an evolving or variant system design. Also, progress of the systems engineering effort is based on the documentation status that may not adequately reflect the system requirements and design quality. These limitations can result in inefficiencies and potential quality issues that often show up during integration and testing, or worse, after the system is delivered to the customer.

2.1.2 Model-Based Systems Engineering Approach

A model-based approach has been standard practice in electrical and mechanical design and other disciplines for many years. Mechanical engineering transitioned from the drawing board to increasingly more sophisticated two-dimensional (2D) and then three-dimensional (3D) computer-aided design tools beginning in the 1980s. Electrical engineering transitioned from manual circuit design to automated schematic capture and circuit analysis in a similar time frame. Computer-aided software engineering became popular in the 1980s for using graphical models to represent software at abstraction levels above the programming language. The use of modeling for software development is becoming more widely adopted, particularly since the advent of the Unified Modeling Language in the 1990s.

The model-based approach is becoming more prevalent in systems engineering. A mathematical formalism for MBSE was introduced in 1993 [24]. The increasing capability of computer processing, storage, and network technology along with emphasis on systems engineering standards has created an opportunity to significantly advance the state of the practice of MBSE. It is expected that MBSE will become standard practice in a similar way that it has with other engineering disciplines.

"**Model-based systems engineering** (**MBSE**) is the formalized application of modeling to support system requirements, design, analysis, verification, and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases" [25]. MBSE is intended to facilitate systems engineering activities that have traditionally been performed using the document-based approach and result in enhanced communications, specification and design precision, system design integration, and reuse of system artifacts. The output of the systems engineering activities is a coherent model of the system (i.e., system model), where the emphasis is placed on evolving and refining the model using model-based methods and tools.

The System Model

The **system model** is generally created using a modeling tool and contained in a model repository. The system model includes system specification, design, analysis, and verification information. The model consists of elements that represent requirements, design elements, test cases, design rationale, and their interrelationships. Figure 2.1 shows the system model as an interconnected set of model elements that represent key system aspects as defined in SysML, including its structure, behavior, parametrics, and requirements.

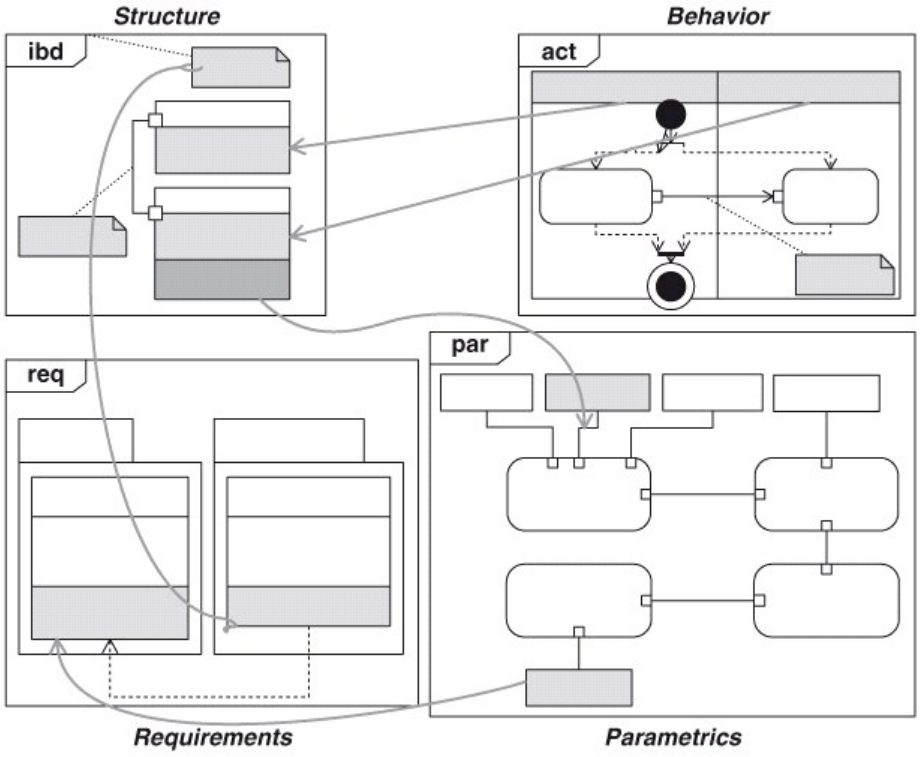

Figure 2.1: Representative system model example in SysML. (Specific model elements have been deliberately obscured and will be discussed in subsequent chapters.)

A primary use of the system model is to design a system that satisfies system requirements and allocates the requirements to the system's components. Figure 2.2 depicts how the system model is used to specify the components of the system.

The system model includes component interconnections and interfaces, component interactions and related functions components must perform, and component performance and physical characteristics. The textual requirements for the components may also be captured in the model and traced to system requirements.

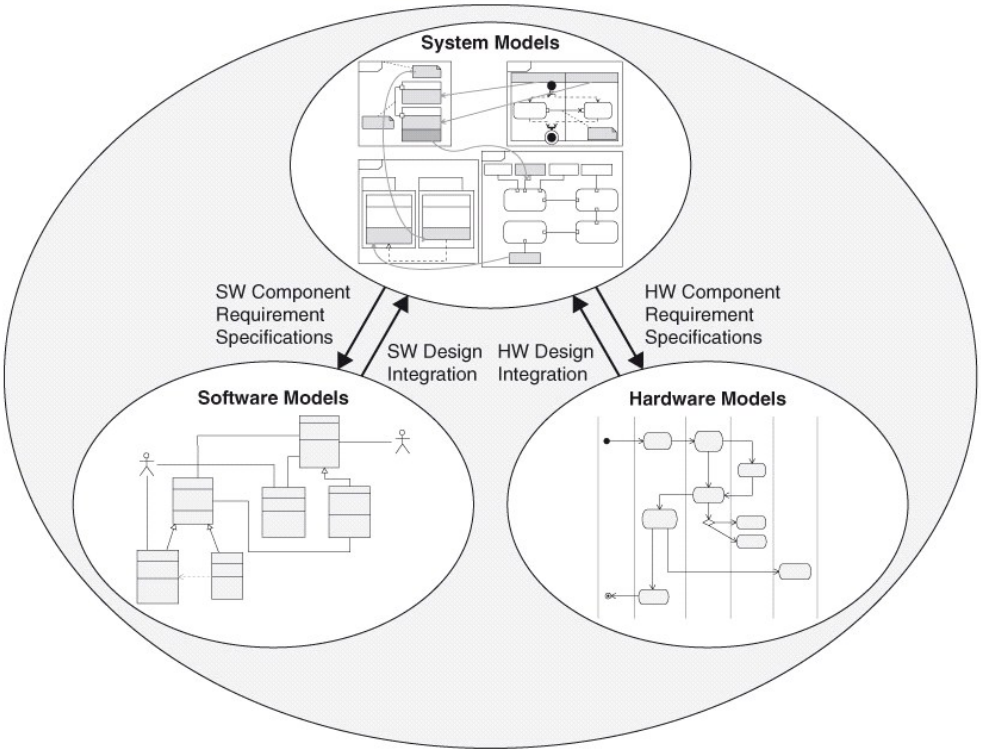

Figure 2.2: The system model is used to specify the components of the system.

In this regard, the system model is used to specify the component requirements and can be used as an agreement between the system designer and the subsystem and or component developer. The component developers receive the component requirements in a way that is meaningful to them either through a model data exchange mechanism or by providing documentation that is automatically generated from the model. The component developer can provide information about how the component design complies with its requirements in a similar way. The use of a system model provides a mechanism to specify and integrate subsystems and components into the system and maintain traceability to higher-level requirements.

The system model can also be integrated with engineering analysis and simulation models to perform computation and dynamic execution. If the system model is executed directly, the system modeling environment must be augmented with an execution environment. A brief discussion of executable models is included in Chapter 17.

The Model Repository

The model elements are stored in a **model repository** and depicted on diagrams by graphical symbols. The tool enables the modeler to create, modify, and delete individual model elements and their relationships in the model repository. The modeler uses the symbols on the diagrams to enter the information into the model repository and to view model repository information. The specification, design, analysis, and verification information previously captured in documents is now captured in the model repository. The model can be viewed in diagrams or tables or in reports generated by querying the model repository. The views enable understanding and analysis of different aspects of the same system model. The documents can continue to serve as an effective means for reporting the information, but in MBSE, the information contained in documentation is generated from the model. In fact, many of the modeling tools have flexible and automated document-generation capability that can significantly reduce the time and cost of building and maintaining the system specification and design documentation.

Model elements corresponding to requirements, design, analysis, and verification information are traceable to one another through their relationships, even if they are represented on different diagrams. For example, an engine component in an automobile system model may have many relationships to other elements in the model. The engine, which is part of the automobile system, is connected to the trans mission, satisfies a power requirement, performs a function to convert fuel to mechanical energy, and has a weight property that contributes to the vehicle's weight.

The static semantics of the model impose rules that constrain which relationships can exist. For example, the model should not allow a requirement to contain a system component or an activity to produce inputs instead of outputs. Additional model constraints may be imposed based on the method being employed. An example of a method-imposed constraint may be that all system functions must be decomposed and allocated to a component of the system. Modeling tools are expected to enforce constraints at the time the model is constructed, or by running a model-checker routine at the modeler's convenience and providing a report of the constraint violations.

The model provides much finer-grain control of the information than is available in a document-based approach, where this information may be spread across many documents and the relationships may not be explicitly defined. The model-based approach promotes rigor in the specification, design, analysis, and verification process. It also significantly enhances the quality and timeliness of traceability and impact assessment over the document-based approach.

Transitioning to MBSE

Models have been used as part of the document-based systems engineering approach for many years, and include functional flow diagrams, behavior diagrams, schematic block diagrams, N2 charts, performance simulations, and reliability models, to name a few. However, the use of models has generally been limited in scope to support specific types of analysis or selected aspects of system design. The individual models have not been integrated into a coherent model of the overall system, and the modeling activities have not been integrated into the systems engineering process. The transition from document-based systems engineering to MBSE is a shift in emphasis from controlling the documentation about the system to controlling the model of the system. MBSE integrates system requirements, design, analysis, and verification models to address multiple aspects of the system in a cohesive manner, rather than a disparate collection of individual models.

MBSE provides an opportunity to address many of the limitations of the document-based approach by providing a more rigorous means for capturing and integrating system requirements, design, analysis, and verification information, and facilitating the maintenance, assessment, and communication of this information across the system's life cycle. Some of the MBSE potential benefits include the following:

- Enhanced communications
  - Shared understanding of the system across the development team and other stakeholders
  - Ability to integrate views of the system from multiple perspectives
- Reduced development risk
  - Ongoing requirements validation and design verification
  - More accurate cost estimates to develop the system
- Improved quality
  - More complete, unambiguous, and verifiable requirements
  - More rigorous traceability between requirements, design, analysis, and testing
  - Enhanced design integrity
- Increased productivity
  - Faster impact analysis of requirements and design changes
  - Reuse of existing models to support design evolution
  - Reduced errors and time during integration and testing
  - Automated document generation
- Enhanced knowledge transfer
  - Specification and design information captured in a standard format that can be accessed via query and retrieval

MBSE can provide additional rigor in the specification and design process when implemented using appropriate methods

and tools. However, this rigor does not come without a price. Clearly, transitioning to MBSE underscores the need for up-front investment in processes, methods, tools, and training. It is expected that during the transition, MBSE is performed in combination with document-based approaches. For example, the upgrade of a large, complex legacy system still relies heavily on the legacy documentation, and only parts of the system may be modeled. Careful tailoring of the approach and scoping of the modeling effort is essential to meet the needs of a particular project. The considerations for transitioning to MBSE are discussed in Chapter 18.

2.2 Modeling Principles

The following sections provide a brief overview of some of the key modeling principles.

2.2.1 Model and MBSE Method Definition

A **model** is a representation of one or more concepts that may be realized in the physical world. It generally describes a domain of interest. A key feature of a model is that it is an abstraction that does not contain all the detail of the modeled entities within the domain of interest. Models are represented in many forms including graphical, mathematical, and logical representations, and physical prototypes. For example, a model of a building may include a blueprint and a scaled prototype physical model. The building blueprint is a specification for one or more buildings that are built. The blueprint is an abstraction that does not contain all the building's detail such as the characteristics of its materials.

A SysML model is analogous to a building blueprint that specifies a system to be implemented. Instead of a geometric representation of the system, the SysML model represents the behavior, structure, properties, constraints, and requirements of the system. SysML has a semantic foundation that specifies the types of model elements and the relationships that can appear in the system model. The model elements that comprise the system model are stored in a model repository and can be represented graphically. A SysML model can also be simulated if it is supported by an execution environment.

A **method** is a set of related activities, techniques, and conventions that implement one or more processes and is generally supported by a set of tools. A **model-based systems engineering method** can be characterized as a method that implements all or part of the systems engineering process, and it produces a system model as one of its primary artifacts.

2.2.2 The Purpose for Modeling a System

The purpose for modeling a system for a particular project must be clearly defined in terms of the expected results of the modeling effort, the stakeholders who use the results, and how the results are intended to be used. The model purpose is used to determine the scope of the modeling effort in terms of model breadth, depth, and fidelity. This scope should be balanced with the available schedule, budget, skill levels, and other resources. Understanding the purpose and scope provides the basis for establishing realistic expectations for the modeling effort. The purposes for modeling a system may emphasize different aspects of the systems engineering process or support other life-cycle uses, including the following:

- Characterize an existing system
- Specify and design a new or modified system
  - Represent a system concept
  - Specify and validate system requirements
  - Synthesize system designs
  - Specify component requirements
  - Maintain requirements traceability
- Evaluate the system
  - Conduct system design trade-offs
  - Analyze system performance requirements or other quality attributes
  - Verify that the system design satisfies its requirements

- Assess the impact of requirements and design changes

- Train users on how to operate or maintain a system

2.2.3 Establishing Criteria to Meet the Model Purpose

Criteria can be established to assess how well a model can meet its modeling purpose. However, one must first distinguish between a good model and a good design. One can have a good model of a poor design or a poor model of a good design. A good model meets its intended purpose. A good design is based on how well the design satisfies its requirements and the extent to which it incorporates quality design principles. As an example, one could have a good model of a chair that meets its intended purpose by providing an accurate representation of the modeled system. However, the chair's design may be a poor design if it does not have structural integrity. A good model provides visibility to aid the design team in identifying issues and assessing design quality. The selected MBSE method and tools should facilitate a skilled team to develop both a good model and a good design.

The answers to the following questions can be used to assess the goodness of the model and derive quality attributes of it. The quality attributes in turn can be used to establish preferred modeling practices.

Is the Model's Scope Sufficient to Meet its Purpose?

Assuming the purpose is clearly defined as described earlier, the scope of the model is defined in terms of its breadth, depth, and fidelity. The model scope significantly impacts the level of resources required to support the modeling effort.

Model breadth. The breadth of the model must be sufficient for the purpose by determining which parts of the system need to be modeled. This question is particularly relevant to large systems where one may not need to model the entire system to meet project needs. If new functionality is being added to an existing system, one may choose to focus on modeling only those portions needed to support the new functionality. In an automobile design, for example, if the emphasis is on new requirements for fuel economy and acceleration, the model may focus on elements related to the power train, with less focus on the braking and steering subsystems.

Model depth. The depth of the model must be sufficient for the purpose by determining the level of the system design hierarchy that the model must encompass. For a conceptual design or initial design iteration, the model may only address a fairly high level of the design. In the automobile example, the initial iterations may only model to the engine level, where a future design iteration may model the engine parts if it is subject to further development.

Model fidelity. The fidelity of the model must be sufficient for the purpose by determining the required level of detail for different modeling constructs. For example, a low-fidelity behavioral model may be sufficient to communicate a simple ordering of actions in an activity diagram. Additional detail is required if the behavioral model is intended to be executed to validate the logic. When modeling interfaces, a low-fidelity model may only include the logical interface description, where as a higher-fidelity model may model the communication protocol. Additional detail is required to model system performance.

Is the Model Complete Relative to its Scope?

A necessary condition for the model to be complete is that its breadth, depth, and fidelity must match its defined scope. Other completion criteria may relate to other quality attributes of the model (e.g., whether the naming conventions have been properly applied) and design completion criteria (e.g., whether all design elements are traced to a requirement). The MBSE metrics discussed in Section 2.2.4 can be used to establish additional completion criteria.

Is the Model Well Formed Such that Model Constraints are Adhered To?

A well-formed model conforms to its static semantics. For example, the static semantics in SysML do not allow a requirement to contain a system component, although other relationships are allowed between components and requirements such as the satisfy relationship. The modeling tool should enforce the constraints imposed by the static semantics or provide a report of violations.

Is the Model Consistent?

In SysML, some rules are built into the language to ensure model consistency. For example, compatibility rules can support type checking to determine whether interfaces are compatible or whether units are consistent on different properties. Additional constraints can be imposed by the method used. For example, a method may impose a constraint that logical components can only be allocated to hardware, software, or operational procedures. These constraints can be expressed in the object constraint language (OCL) [26] and enforced by the modeling tool.

Enforcing constraints assists in maintaining consistency across the model, but it does not prevent inconsistencies. A simple example may be that a modeler inadvertently gives a component two different names that are interpreted by a model checker as different components. The likelihood of inconsistencies increases when multiple people are working on the model. A combination of well-defined model conventions and a disciplined process can limit this from happening.

Is the Model Understandable?

There are many factors driven by the model-based method and modeling style that can contribute to understandability. A key contributing factor to enhance understandability is the effective use of model abstraction. For example, when describing the functionality of an automobile, one could describe a top-level function as "drive car" or provide a more detailed functional description such as "turn ignition on, put gear into drive, push accelerator pedal," and so on. An understandable model should include multiple levels of abstraction that represent different levels of detail but relate to one another. As will be described in later chapters, the use of decomposition, specialization, allocations, views, and other modeling approaches in SysML can be used to represent different levels of abstraction.

Another factor that impacts understandability relates to the presentation of information on the diagrams themselves. Often, there is a lot of detail in the model, but only selected information is relevant to communicate a particular design aspect. The information on the diagram can be controlled by using the tool capability to elide (hide) nonessential information and display only the information relevant to the diagram's purpose. Again, the goal is to avoid information overload for the reviewer of the model.

Other factors that contribute to understandability are the use of modeling conventions and the extent to which the model is self-documenting as described next.

Are Modeling Conventions Documented and Used Consistently?

Modeling conventions and standards are critical to ensure consistent representation and style across the model. This includes establishing naming conventions for each type of model element, diagram names, and diagram content. Naming conventions may include stylistic aspects of the language, such as when to use uppercase versus lowercase, and when to use spaces in names. The conventions and standards should also account for tool-imposed constraints, such as limitations in the use of alphanumeric and special characters. It is also recommended that a template be established for each diagram type so that consistent style can be applied.

Is the Model Self-Documenting in Terms of Providing Sufficient Supporting Information?

The use of annotations and descriptions throughout the model can help to provide value-added information if applied consistently. This can include the rationale for design decisions, flagging issues or problem areas for resolution, and providing additional textual descriptions for model elements. This enables longer-term maintenance of the model and enables it to be more effectively communicated to others.

Does the Model Integrate with Other Models?

The system model may need to be integrated with electrical, mechanical, software, test, and engineering analysis models. This capability is determined by the specific method, tool implementation, and modeling languages used. For example, the approach for passing information from the system model using SysML to a software model using UML can be defined for specific methods and tools. In general, this is addressed by establishing an agreed-on expression of the modeling information so that it can be best communicated to the user of the information, such as hardware and software developers, testers, and engineering analysts.

2.2.4 Model-Based Metrics

Measurement data collection, analysis, and reporting can be used as a management technique throughout the development process to assess design quality and progress. This in turn is used to assess status and risk and to support ongoing project planning and control. Model-based metrics can provide useful data that can be derived from the model and can help answer the following questions. This discussion refers to metrics that can be derived from a typical SysML model.

What is the Quality of the Design?

Metrics can be defined to measure the quality of a model-based system design based on metrics that have been traditionally used in document-centric designs. This includes metrics for assessing requirements satisfaction, critical performance properties, and how well the design is partitioned.

A SysML model can provide explicit relationships that can be used to measure the extent that the requirements are

satisfied. The model can provide granularity by identifying model elements that satisfy specific requirements. The requirements traceability can be established from mission-level requirements down to component-level requirements. Other SysML relationships can be used in a similar way to measure which requirements have been verified. These data can be captured directly from the model or indirectly from a requirements management tool that is integrated with the SysML modeling tool.

A SysML model can include critical properties that are monitored throughout the design process. Typical properties may include performance properties, such as latency, physical properties (e.g., weight), and other properties (e.g., reliability and cost). These properties can be monitored using standard technical performance measurement (TPM) techniques. The model can also include relationships among the properties that indicate how they may be impacted as a result of design decisions.

Design partitioning can be measured in terms of the level of cohesion and coupling of the design. Coupling can be measured in terms of the number of interfaces or in terms of more complex measures of dependencies between different model parts. Cohesion metrics are more difficult to define, but measure the extent to which a component can perform its functions without requiring access to external data. The object-oriented concept of encapsulation reflects this concept.

What is the Progress of the Design and Development Effort?

Model-based metrics can be defined to assess design progress by establishing completion criteria for the design. The quality attributes in the previous section referred to whether the model is complete relative to the defined scope of the modeling effort. This is necessary, but not sufficient, to assess design completeness. The requirements satisfaction described to measure design quality can also be used to assess design completeness. Other intermediate metrics may include the number of use case scenarios that have been completed or the percent of logical components that have been allocated to physical components. From a systems engineering perspective, a key measure of system design completeness is the extent to which components have been specified. This metric can be measured in terms of the completeness of the specification of component interfaces, behavior, and properties.

Other metrics for assessing progress include the extent to which components have been verified and integrated into the system, and the extent to which the system has been verified to satisfy its requirements. Test cases and verification status can be captured in the model and used as a basis for this assessment.

What is the Estimated Effort to Complete Design and Development?

The Constructive Systems Engineering Cost Model (COSYSMO) is used for estimating the cost and effort to perform systems engineering activities. This model includes both sizing and productivity parameters, where the size estimates the magnitude of the effort, and productivity factors are applied to come up with an actual labor estimate to do the work.

When using model-based approaches, sizing parameters can be identified in the model in terms of numbers of different modeling constructs that may include the following:

Requirements

Use cases

Scenarios

States

System and component interfaces

System and component activities or operations

System and component properties

Components by type (e.g., hardware, software, data, operational procedures)

Test cases

The MBSE sizing parameters will need to be integrated into the cost model. Data will need to be collected and validated over time to establish statistically meaningful data. However, early users of MBSE can identify sizing parameters that contribute most significantly to the modeling effort, and use this data for local estimates and to assess productivity improvements over time.

2.2.5 Other Model-Based Metrics

The previous discussion is a sampling of some of the model-based metrics that can be defined. Many other metrics can also be derived from the model, such as the stability of the number of requirements and design changes over time, or potential defect rates. The metrics can also be derived to establish benchmarks from which to measure the MBSE benefits as described in Section 2.1.2, such as the productivity improvements resulting from MBSE over time. Chapter 18 includes a discussion of additional organizational metrics related to deploying MBSE in an organization.

2.3 Summary

The practice of systems engineering is transitioning from a document-based approach to a model-based approach like many of the other engineering disciplines, such as mechanical and electrical engineering, have already done. MBSE offers significant potential benefits to enhance communications, specification and design precision, design integration, and reuse that can improve design quality, productivity, and reduce development risk. The emphasis in MBSE is on producing and controlling a coherent system model, and using this model to specify and design the system. Quality attributes of a model such as model consistency, understandability, and well formedness, and the use of modeling conventions, can be used to assess the goodness of a model and to derive preferred modeling practices. MBSE metrics can be used to assess design quality, progress and risk, and support management of the development effort.

2.4 Questions

1. What are some of the primary distinctions between MBSE and a document-based approach?
2. What are some of the benefits of MBSE over the document-based approach?
3. Where are the model elements of a system model stored?
4. Which aspects of the model can be used to define the scope of the model?
5. What constitutes a good model?
6. What are some of the quality attributes of a good model?
7. What is the difference between a good model and a good design?
8. What are examples of questions that MBSE metrics can help answer?
9. What are possible sizing parameters that could be used to estimate an MBSE effort?