The Cascade – Building a Medical Device

TGFR Consulting LLC

September 2018

Contents

1 Introduction

This book is not intended to provide the reader a compendium of methods and tools that can be used during the development of a medical device. What this book does present is a workflow, where at each step in the design and development process, specific techniques and tools can be employed to make the results of the step more complete as well as simplifying the subsequent steps in the development.

The following are several simple examples of how the techniques described in this book integrate the process of medical device development.

- The use of fishbone diagrams and techniques to identify the features (columns) of a QFD (Qualify Function Deployment) can significantly improve the primary purpose of the QFD, the identification of the key “critical-to-quality features” (CTQs) that will ensure meeting the user needs
- Linking the top-down analysis of the therapy work flow ensures that the use cases developed as part of the requirements clearly link to the hazard analysis. Using these use case to build the validation scenarios, this linkage can ensure that use errors uncovered during scenario execution can correctly be assessed against the risk profile of the device.
- Consideration of the system architecture during the development of the Design Inputs (system requirements) can significantly lower the overall verification effort. Requirements can be better allocated to the subsystems and coupling that could introduce additional testing during verification can be avoided.

The workflow presented drives the minimization of iterations between steps. While many of the steps include iteration within the step, the workflow eliminates the iteration between major steps. The presence of “doom loops” in a workflow creates iteration and rework between steps delaying the effort and causing quality problems (Pugh, 1981). This workflow and techniques drive the elimination of the “undiscovered rework” associated with incomplete work outputs. Each step has a clear set of deliverables that do not require subsequent updates. Completion in the context of this workflow means the following

- Each output provides the necessary information for subsequent steps in the workflow
- The documentation of a workflow steps is sufficient. The required artifacts can be clearly mapped to the artifacts associated with regulated medical device development

2 Overview

2.1 Creating the Requirements

The process of creating the requirements for the project starts with the Voice of the Customer (VOC) and the Voice of the Business (VOB).

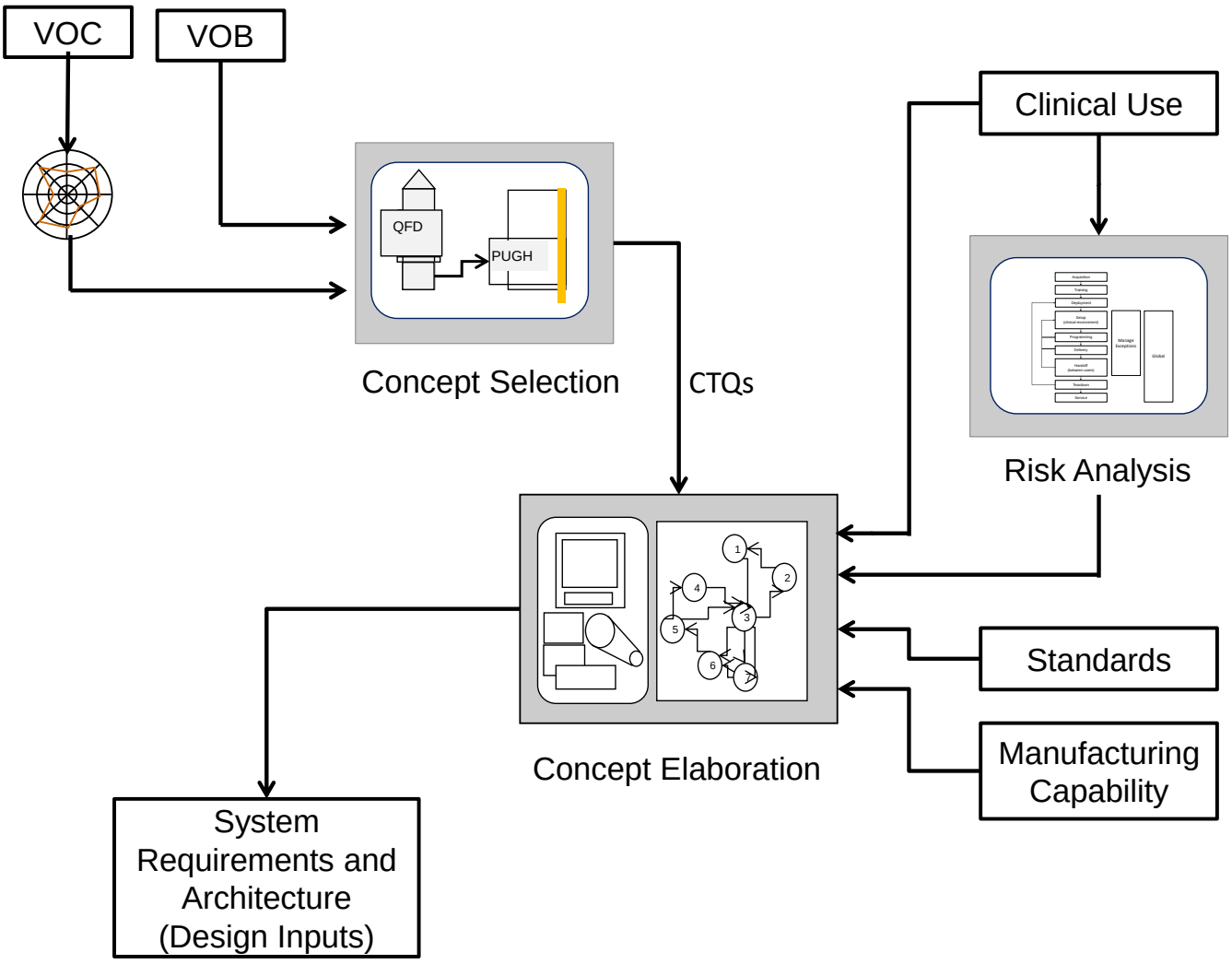


Figure 2-1 Creating the Requirements

Using the VOC and the VOB concept selection uses

- The QFD (Quality Function Deployment) to develop the Critical to Quality (CTQ) Features
- The Pugh Matrix and Super Concept to determine the final Concept

Based upon the final concept, development turns to the process of elaborating the concept into a final set of requirements that characterize the product and guide all further development. At this stage in the workflow, the elaboration process focuses on the development of the full range of system level the requirements incorporating all of the various sources (CTQs, Regulatory Standards, Risk and Manufacturability).

The final step of the elaboration step confirms the final concept with the customers, users and

the business. With the confirmation of the concept, architecture and associated requirements, the process can move on to development.

The requirements development process leverages the following tools and techniques

- User Needs Statements
- The Spider Chart
- The QFD
- The Pugh Matrix
- Risk Analysis

2.2 Developing the Product

Developing the product transforms the concept, architecture and system requirements into the medical device, which after validation represents a finished product.

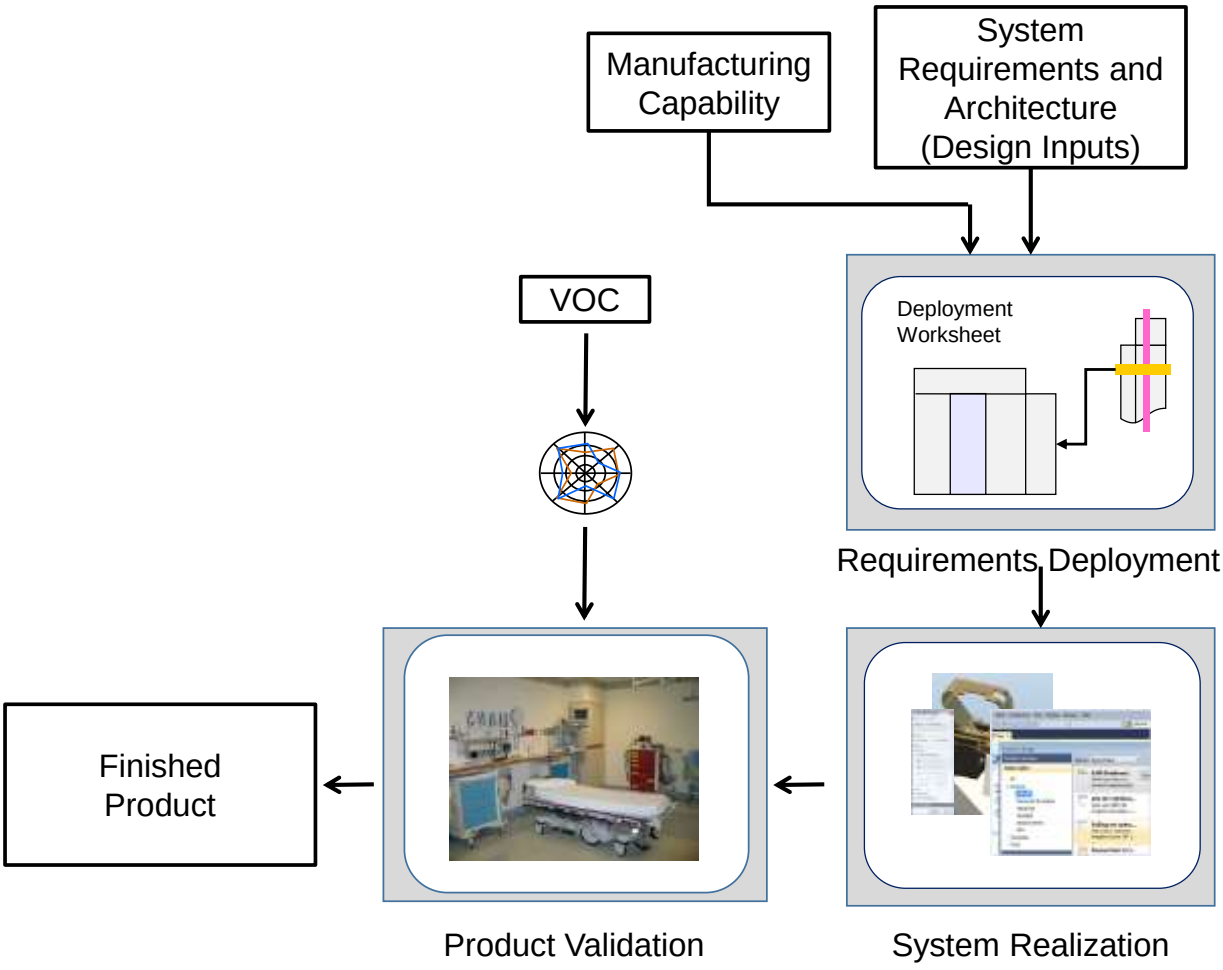


Figure 2-2 Developing the Product

Deployment uses the architecture to partition and flow down the system requirements to the subsystems. The deployment worksheets created connect the elements of the system and provide the framework for system realization. System realization consists of detailed subsystem design and development, system integration and the final formal verification of the design. Upon completion of realization, design validation confirms that the device meets the original Voice of the Customer (VOC) and user needs, as set forth during the requirements development activities.

3 Creating the Requirements

3.1 Concept Selection

The following table describes the inputs and outputs of the Concept Selection Process

Table 3-1 Concept Selection SIPOC

Inputs	Key Activities	Outputs
Seminal Idea Clear statement of the basic target function **VOC – The Customer Requirements** The expression of the effect that a customer would like to have on his life/environment to best perform his desired tasks **VOB – Business Requirements** The Needs of the business	**VOC Development** Building the user need statements **Feature Deployment** Using the user needs to develop the features (columns of the QFD) **QFD** Identifying the Critical to Quality Features **Pugh Matrix and Super Concepts** The generation of the overall concept and adjustment of the CTQs (Critical to Qualify) **Early Concept Confirmation** The initial confirmation of the concept with the users	System Concept Definition The concept for the System **CTQs and Other Features** The key feature requirements for the concept

3.1.1 VOC Development

VOC (voice of the customer) is an expression of the effect that a customer would like to have on his life/environment for him to best perform his desired tasks. VOC development includes

- The process of gathering, understanding, targeting, and verifying the needs of non-business ‘actors’ directly affiliated with the entities that purchase the product.
- Understanding and targeting of the needs of paying customers (customer defined quality), both obvious and latent, and will drive product’s competitive advantage in the market.

Simply put, the customer includes the users of the product and those that pay for the product. Each group has needs and these needs must be identified and characterized. VOC should be the structured framework that expresses these needs.

The development starts with customer engagements and interviews to collect information and statements that will become VOC. Most customers, when interviewed, state their needs in terms of solutions to the last problem encountered. As part of the user needs collection, conscious steps must be taken to identify statements that represent solutions. The VOC collection step transforms these customer solution statements to true user needs.

Identification of solutions statements, as opposed to a statement of need, represents the first step in the process of collecting user needs. Statements of a solution often involve the word “should” or “must”, indicating that the statement is describing the behavior of the system. The recognition of a solution statement from the customer should prompt the developer/interviewer to probe further with the customer to identify the true need.

The “five-why” tool (Wilson, n.d.) facilitates driving to the true need. This iterative, question-asking technique explores the cause-and-effect relationships underlying a particular problem statement. While this technique was originally developed to determine the root cause of a problem, the basic principle of repeatedly driving answers to a supporting “why” question works effectively to determine a true user need statement. Each question forms the basis of the next question. The following table demonstrates the application of the "five-why" tool.

Table 3-2 Five-Why Process

User Statement	Question
I want the system to deliver a weight based dose	Why do you want a weight based dose?
The dosing comes from the pharmacy as a weight based dose	Why does it come from the pharmacy as a weight based dose?
Because the Doctor Prescribes the dose as weight based	

With this series of questions, successive “whys” have provided the true user need as the following statement:

I need the patient to receive the exact prescription as ordered by the doctor.

Following the collection process, the needs statements need to be evaluated and structured as effective user need statements. The following are good rules for an effective user need statement (Bettencourt, Spring 2008)

- **The statement must reflect the customer's definition of value** – value must be defined and measured from the customer's perspective
- **The statement must have universal acceptance** – it should be relevant to all customers
- **The statement must be relevant now and in the future**
- **The statement must prompt a course of action** – statements such as "it must be more reliable" or "easy to use" will not define a course of action.
- **The statement must not be open to interpretation**
- **The statement itself shall not confound the way it or other statements are prioritized** – be careful that the statements do not overlap.

In order to ensure that the collected statements do conflict or overlap, the collected statements should be reduced to a succinct set of 10-15 user needs. More than 10-15 user need statements will impact subsequent steps that rely on the user needs, diluting the process of identifying the critical parameters for a successful design. After the interview process, the user need statements should go through a consolidation pass to reach a final list a 10-15 distinct statements.

An affinity exercise (Tague, 2004) effectively consolidates user need statements. In an affinity exercise, similar statements are grouped on a whiteboard or wall. These grouping are then consolidated into a single, summary user need statement. These summary user need statements become the VOC.

Following the development of the user needs statement, a scoring process engages customers and users to score current product performance as well as establish a score for best-in-class performance against the user need statements. Scoring should use a Likert scale (Vanek, 2012). The scale should be a 1-10 range, with 10 as the maximum. Following scoring, compile the scaled results into a radar chart mapping needs to performance, as shown below.

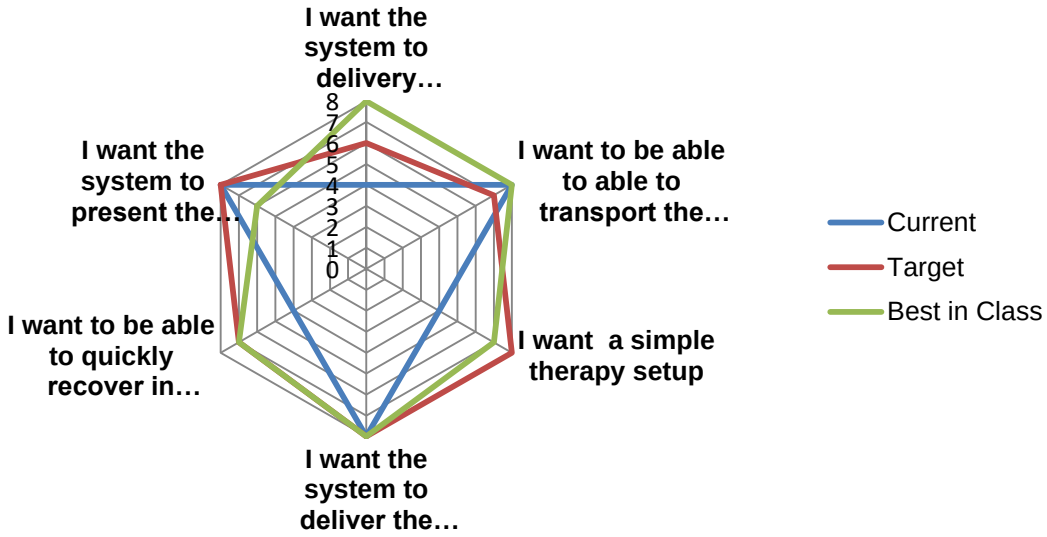


Figure 3-1 Radar Chart Mapping User Needs

In situations where competition (or lack of competition) may impact collecting best-in-class scores, a different best-in-class scoring technique can be applied. In this situation, the user should be asked to provide the top score against a user need by considering any piece of equipment in the use area. With this technique, it is very likely that the best-in-class score for each user need will be associated with a different device. This technique will allow devices without competition to find an applicable best-in-class score.

VOC represents the 10-15 user need statements that drive all subsequent development. In summary, VOC provides the following

- **An understanding of what matters to the customer** – Clear, concise statements of needs
- **What represents "best in class"** - measured against the real user needs
- **Where the current product stands** - again, measured against the real user needs

3.1.2 VOB

VOB (voice of the business) represents the commercial and other business considerations for the medical device. VOB development must be kept distinct from the activities associated with user needs in order to prevent the corruption of the true user needs. In developing these distinct needs, every effort should be made to make the statement of these needs as quantifiable as possible, as this will be a critical aspect of the final concept evaluation. As an example,

The system shall enable a gross margin of 65%

This statement represents a quantifiable business need that can be easily used to evaluate concepts for the medical device. The following rules apply to the development of VOB:

- **The statement should be quantifiable** - this will drive better evaluations in subsequent steps.
- **The statement must have universal acceptance** – it should be agreed upon throughout the business
- **The statement must prompt a course of action** – statements such as “it must be more aesthetically pleasing” will not define a course of action.
- **The statement itself shall not confound the way it or other statements are prioritized** – be careful that the statements do not overlap.

These rules represent a subset of the rules for customer input, with the business representing the customer.

Key to the development of VOB is the concept that the statement should be quantifiable. Business needs, as a result of being closely associated with the development, can drive a number of qualitative needs. These should be avoided, as these needs often represent the business interpreting customer needs. By focusing on the quantifiable, VOB can be focused on true business needs.

3.1.3 Feature Development and Performance Parameters

Feature Deployment maps the VOC statements to a set of device features to be used as part of the QFD process. An effective mapping effort creates a minimum set of performance parameters, sufficient to allow the QFD to be effective. Developing the minimum set of performance parameters drives more effective execution of the QFD process. Populating the columns of the QFD with this minimum set of parameters simplifies QFD execution and the

subsequent results do a much better job identifying the key CTQs (critical to quality features) for the product.

Specifically, a performance parameter adheres to the following definition

> *A performance parameter represents the definition of a tangible/measurable deliverable of the future product.*

Performance parameters are measurable. Performance parameters have a range of values. The ability to vary the range of a parameter represents the key discriminant for performance parameters. A required feature may be important, but without the ability to vary over a range this feature cannot bet a performance parameter to be be used in QFD process. This discrimination can be summarized as the following key concept

> *All performance parameters are features, but not all features are performance parameters.*

In developing performance parameters, care must be taken not to assume a particular implementation. At this point in the process, a future product concept has yet to be defined. Features should not assume anything related to implementation. A key tool that can help with the development of an implementation-free set of parameters is the standard fishbone (Fishbone (Ishikawa) Diagram, n.d.) diagram. Use of the fishbone diagram facilitates the general mapping from a user need (the effect) to the features or performance parameters that will be used in the QFD process. The fishbone provides the framework and structure around determining how features contribute to the user need. A traditional fishbone contains the following generic groups of contributing features

- **Methods** – The use cases associated with the use of the product
- **Associated Equipment** – other equipment associated with the use of the product.
- **User** – the skills and training of the users
- **Materials** – the material involved in the device
- **Measurement** – the accuracy of the device during operation
- **Environment** – this would be the operating environment for the product

These categories make up the branches of the fishbone, and often these generic grouping can help push the brainstorming to consider all areas where features may impact the user need. Care should be taken to limit the brainstorming to these general categories, as often other features that don't easily fit into these generic groups will be identified.

This fishbone process allows the identification of the minimal number of features that influence satisfaction of the VOCs. Only those features directly contributing to the VOC are identified, with extraneous features ignored. While other features may be identified later in the requirements design, only features directly influencing user needs should be considered for use in the QFD process. Features that are attributes, without the ability to have a range of values can be identified and omitted from the final list of performance parameters. Attribute

features will be important in the development of the final concept or design, but these features do not serve as discriminating parameters for the QFD (more on this later).

The following example shows a partial fishbone diagram associated with a VOC need

I need the patient to receive the exact prescription ordered by the doctor

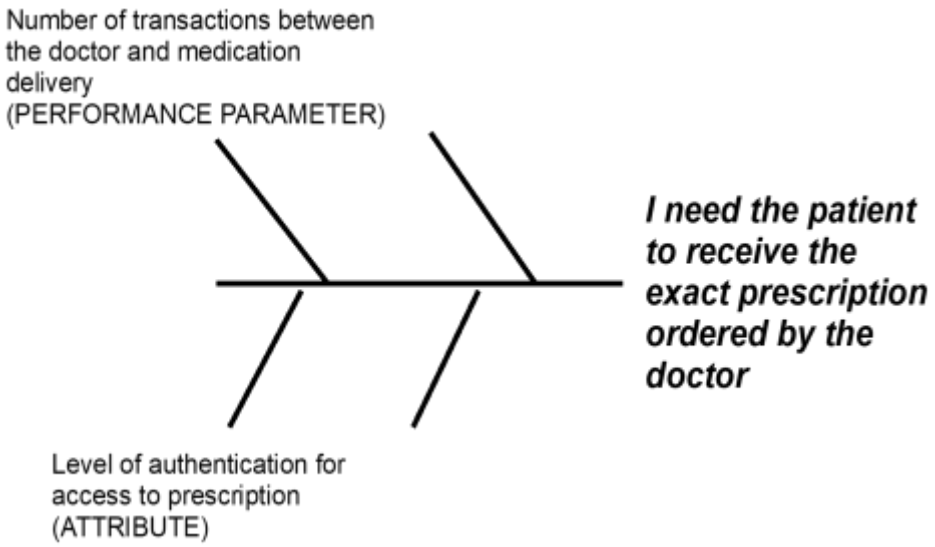


Figure 3-2 Fishbone for User Needs

In this example, two features, “Number of transactions between doctor and medication delivery” and “level of authentication for access to prescription” have been identified as part of the contributing features. The “level of authentication for access to prescription” represents an attribute, as the parameter cannot be varied due to regulatory considerations. The performance parameter is “number of transactions” and the measureable range would be between 1 to 4 transactions.

Following the fishbone drawings/analysis for all of the VOC statements, the resulting full set of performance parameters are coalesced into a single, unique list. Often a single performance parameter contributes to a number of VOC statements, and in coalescing the list all duplicates should be removed. During the QFD process the impact of a single performance parameter will be reflected in the values within the QFD matrix.

3.1.4 QFD (Quality Function Deployment)

Following the identification of the performance parameters, a QFD is constructed and evaluated. A number of books have been written about the QFD process and the level of complexity and detail associated with the QFD execution can be subject to a great deal of debate. The process detailed here will focus on performance parameter identification, and many of the other potential outputs of the process will be ignored. The basic structure of the

QFD is shown below

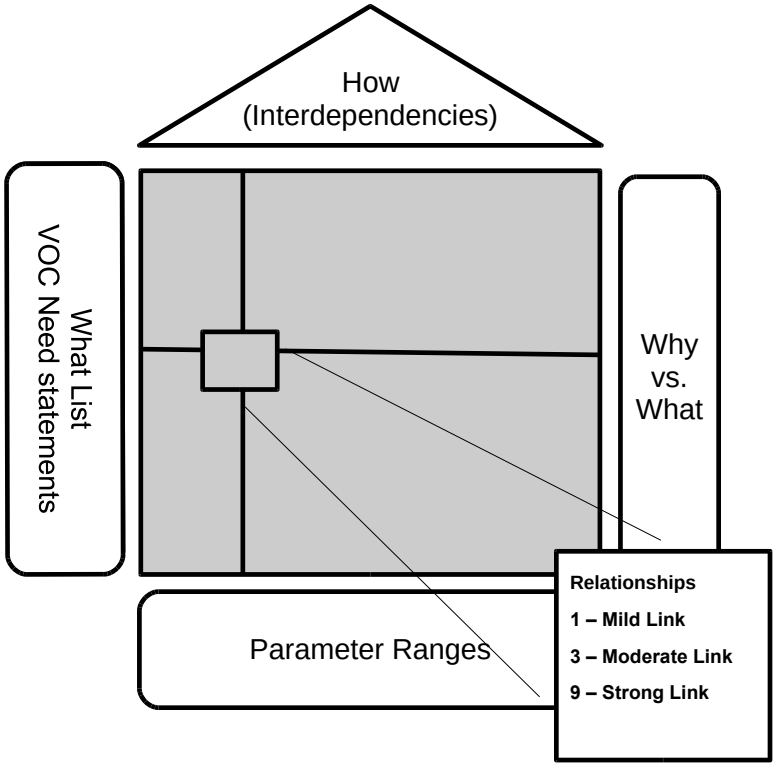


Figure 3-3 QFD Structure

For the QFD, and the mapping of the elements is as follows

- **What List (rows)** – the user needs (VOC). At this point in the process, the VOB is not considered.
- **How (columns)** – the features as developed. Initial ranges are developed.

The preferred structure of the QFD for CTQ identification is that every row has an importance rating, based upon a 1/3/9 rating scale. This 1/3/9 importance rating allows the easy discrimination of the key user needs. The use of a linear 1-5 scale is not recommended. With a linear scale, importance is compressed (everything tends to the same score) and discrimination is lost. With a 1/3/9 scale, the weighting favors the most important, ,but care should be taken to limit the “9” weighting to 1-2 user needs. Otherwise the “Lake Wobegon” effect (all the children are above average) takes over and discrimination is lost.

Scoring the interactions of the features against the user needs scoring also uses 1/3/9 scale. The following definitions should be applied

- 9 - This performance parameter (feature) could, on its own, deliver 75%-100% on this issue/need
- 3- This performance parameter (feature) could, on its own, deliver 25%-75% on this issues/need. Other features will be needed to fully deliver on this issue/need
- 1 - This performance parameter (feature) has an effect of less than 25% on delivery of this issue/need
- Blank - This feature (performance parameter)has no effect on how this issue/need is satisfied

The standard QFD process involves moving the value of a performance parameter over its range and evaluating the changes in its ability to deliver on the user needs. Upon completion of the process the most important performance parameters are identified, and these parameters are the CTQs. A QFD/CTQ process which results in identifying more than one or two CTQs may require adjustments to the weighting and interaction scoring. The identification of too many CTQs usually adversely impacts the selection of a concept and subsequent tradeoffs. In this situation where the QFD process identifies too many CTQs, re-evaluation of the user need ratings and interaction scoring should be undertaken to determine reduce the set of CTQs identified.

As an example of a situation of conflicting CTQs, consider a QFD process for cell phones and other portable devices. In considering the CTQs for cell phones and other portable devices, the CTQs of light weight and long battery life tend to be in conflict. Review and re-execution of the CTQ process forces determination of exactly which of these performance parameters represents the CTQ.

The QFD process develops the CTQs for the device, but the results must represent a set of identified CTQs that the team and the business can agree represent the critical features. These CTQs must be clear and without conflicts that will impact the later concept selection activities.

3.1.5 Concept Development using Super Concepts

Evaluating specific concepts leverages the CTQs and VOB to determine the best overall concept. The evaluation uses a iterative approach based upon several iterations of a Pugh Matrix. The Pugh Matrix scores alternative concepts against a weighted set of criteria. The Pugh Matrix criteria are based upon the CTQs and the VOB established during previous steps. These criteria become the weighted rows of the Pugh Matrix, and the alternative concepts become the columns. Rows are weighted using a 1/3/9 scale as with the QFD based upon the following criteria.

- Only 1 or 2 of the assessment criteria is assigned a value of 9
- The total weight of the rows associated with the CTQs account for 70% of the total weight across all rows
- The total weight of the VOB account no more than 30% of the total weight

Adherence to the guidelines for the total weight assigned to the CTQs versus the total weight

assigned to the VOB is critical. Weighting the VOB rows in the Pugh Matrix too heavily creates the risk that the final concept will meet the VOB at the expense of the user needs as reflected in the CTQs. Customer needs drive the CTQs, and a weighting distribution without a bias towards these user needs may result in the selection of a concept that meets the business needs at the expense of the customer needs. In this situation the process may have a selected a great product that will not appeal to customers.

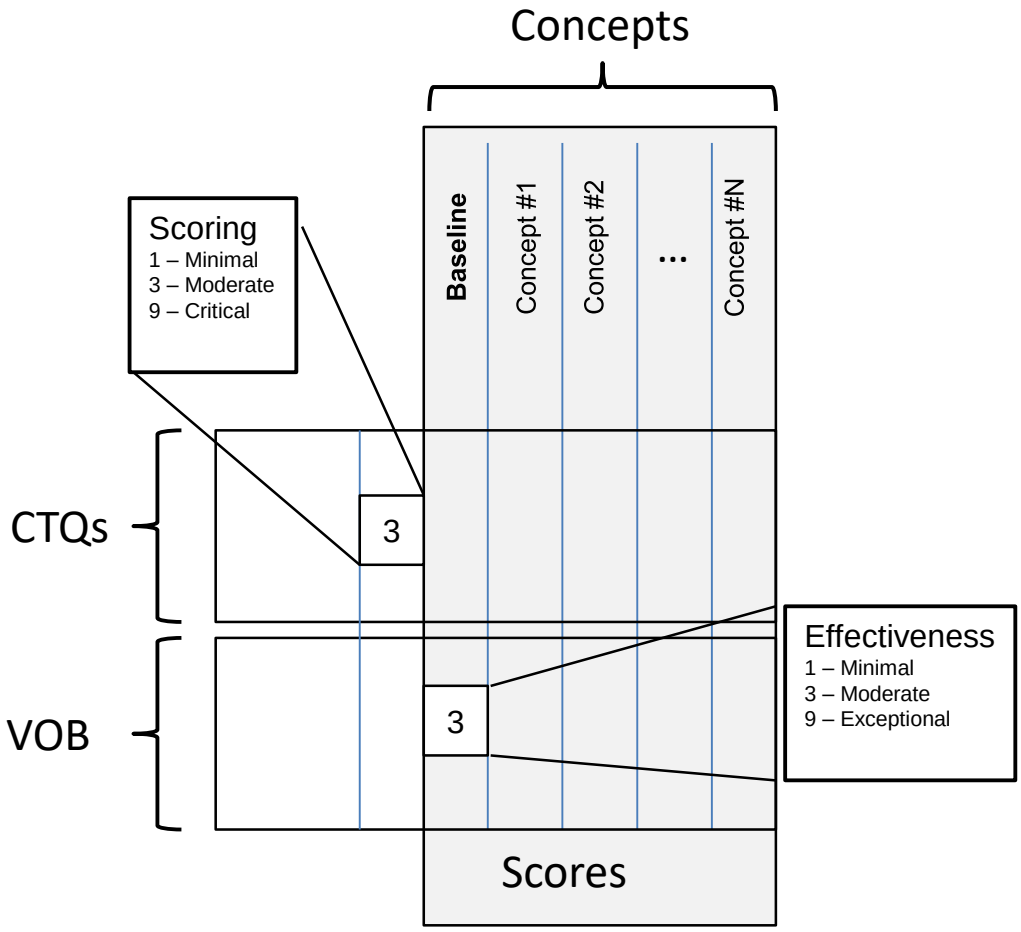


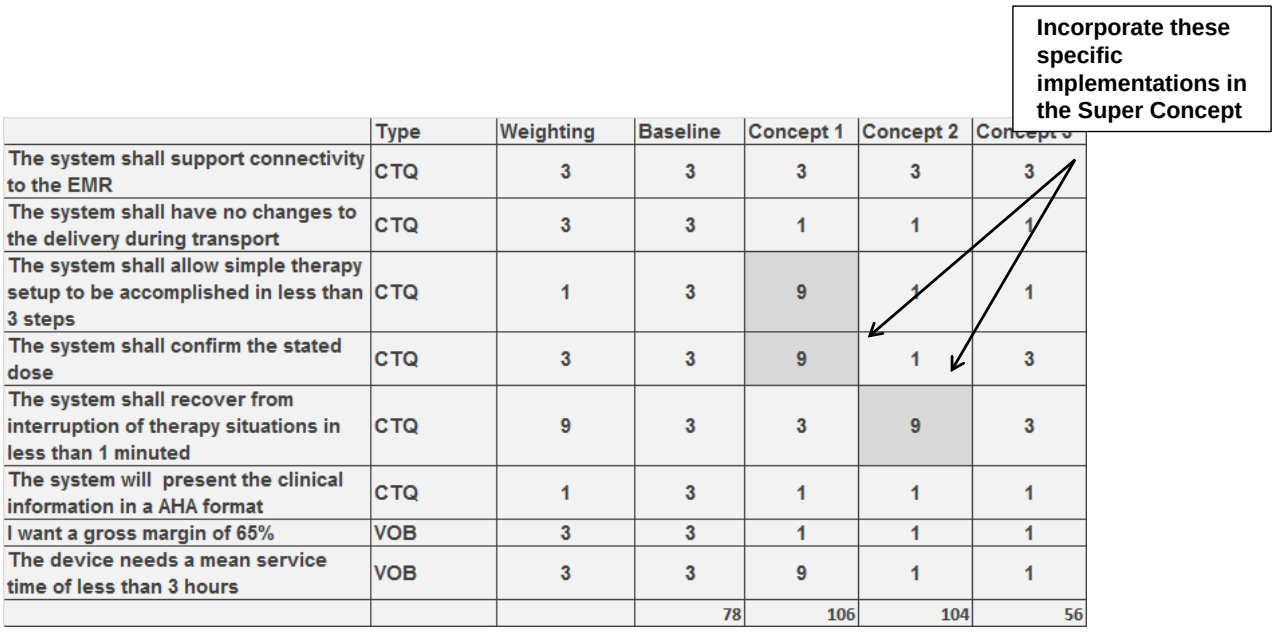
Figure 3-4 Pugh Matrix Structure

Pugh Matrix scoring also follows the 1/3/9 value scale. While many Pugh Matrix discussions favor a +/-/0 subjective scaling, working with numbers and scores often provides a easier framework for the evaluation team to understand.

Concepts, whenever possible, should be developed by competing teams to drive diverse thinking. Following the development and detailing of the concept, the competing teams should get together and collectively score the concepts. This process will build understanding and drive the associated Super Concept process.

The Super Concept process forces at least one round of synthesis, where the best features of each proposed concept are combined into a Super Concept for the next iteration of the concept generation process. The following shows how Super Concept features are driven out

of the evaluation process



	Type	Weighting	Baseline	Concept 1	Concept 2	Concept 3
The system shall support connectivity to the EMR	CTQ	3	3	3	3	3
The system shall have no changes to the delivery during transport	CTQ	3	3	1	1	1
The system shall allow simple therapy setup to be accomplished in less than 3 steps	CTQ	1	3	9	1	1
The system shall confirm the stated dose	CTQ	3	3	9	1	3
The system shall recover from interruption of therapy situations in less than 1 minuted	CTQ	9	3	3	9	3
The system will present the clinical information in a AHA format	CTQ	1	3	1	1	1
I want a gross margin of 65%	VOB	3	3	1	1	1
The device needs a mean service time of less than 3 hours	VOB	3	3	9	1	1
			78	106	104	56

Figure 3-5 Super Concept Identification

To achieve the best concept, several iterations of concept development and scoring should be undertaken. At the point where the teams consider the differences in scoring insignificant, the scoring within the Pugh Matrix is completed and the final concept identified.

3.1.6 Early Concept Confirmation

Concept confirmation follows identification of the final concept. In this step, the concept is realized as drawings, models or other tangible artifacts, and feedback from users and customers gathered to confirm that the concept developed truly meets the user and customer needs. Throughout the concept development process, user and customer needs have been transformed, and this step validates that these transformations resulted in a final concept that meets the original user and customer needs

The confirmation process requires that the fidelity of the concept presented to the users and customers and users allows the users and customers to users to properly evaluate the final concept. Customers are presented with the realized concept and score the concept against the user needs statements and VOC (section 3.1.1VOC Development). The users and customers assess the final concept against the same user needs statement. This confirms the system; understanding, targeting, and verifying the system concept against the needs of users directly affiliated with the entities that purchase the product

Generally, confirmation is against the system derived from the final concept, as defined by the requirements. For some concepts, it may be beneficial to perform a play back the system concept before beginning requirements generation. An early play back of the Super Concept

output, when some uncertainty exists, can avoid churn and rework during requirements generation. At this point, the painstaking work of requirements elaboration has not started and this early concept confirmation can minimize the rework of the requirements because of concept misses.

To effectively evaluate the final concept, a set of questions based upon the Likert scale developed and presented to during the VOC development phase (section 3.1.1, VOC Development).

The figure below illustrates a radar chart showing the customers evaluation of the concept against the original needs, with the target and best-in-class scoring as benchmarks. Concept scores can be compared to the initial target and the best-in-class scores initially developed. In this case, some of the targets have not been met, and the team must decide whether to adjust the target or to undertake another round of concept development. Many times, the target will be adjusted, but this takes place in a structured manner and the CTQs and concepts are re-evaluated against the revised target.

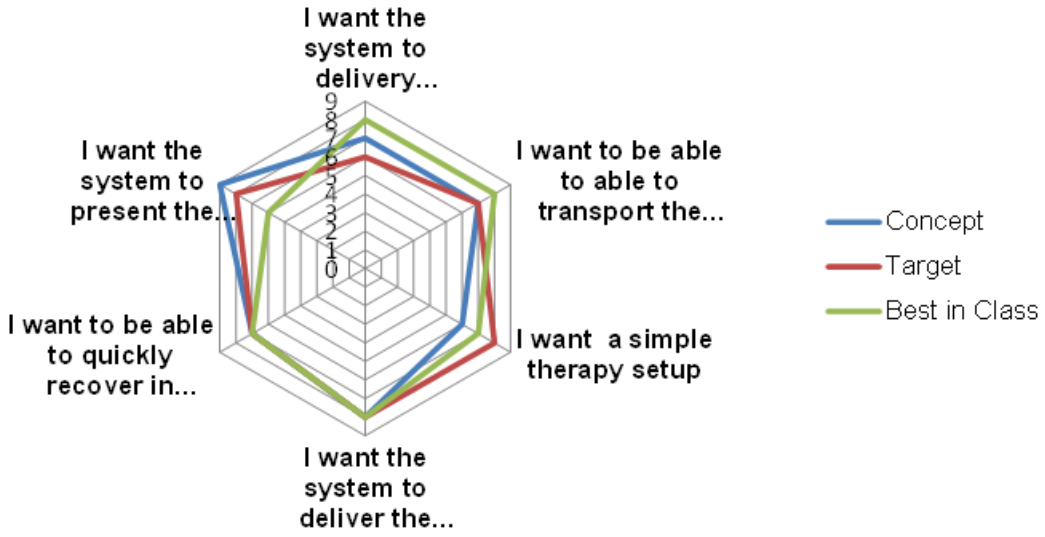


Figure 3-6 Early Concept Confirmation

3.2 Risk Analysis

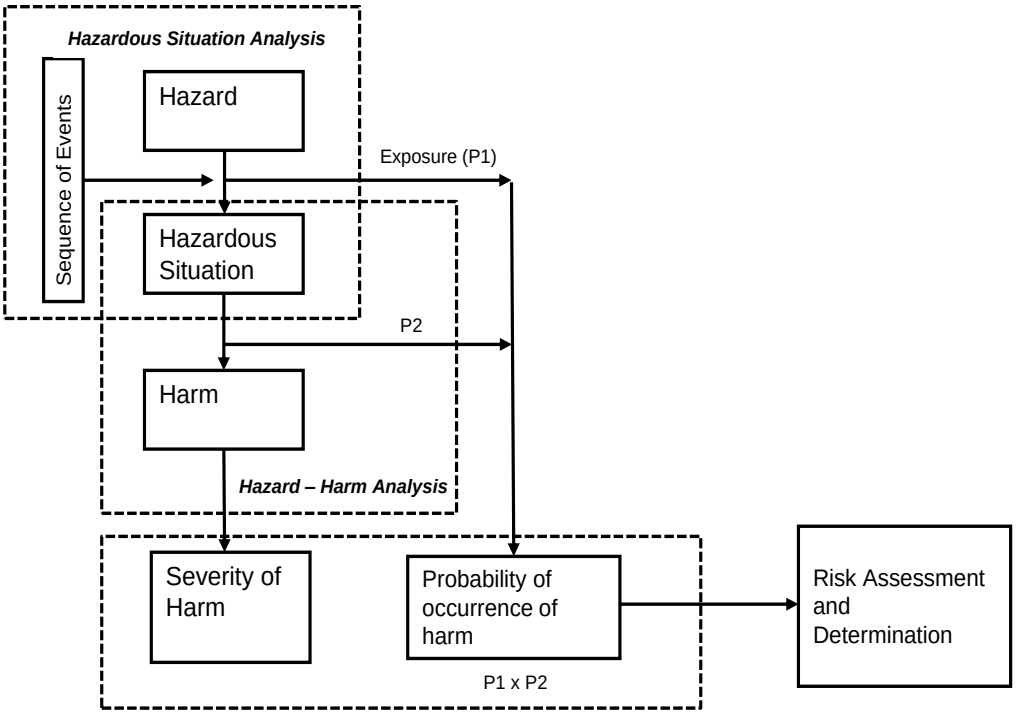
Risk analysis based upon the clinical use of the device must be performed prior to concept elaboration (the development of design inputs). Risk Analysis establishes essential requirements, the requirements associated with safety. These essential requirements will be blended with the concept realization to form the basis of the final design inputs.

The following table describes the inputs and outputs of the Risk Analysis Process

Table 3-3 Risk Analysis SIPOC

Inputs	Key Activities	Outputs
Clinical Use The clinical use of the device **Clinical Use Error Data** Data on the use errors and rates associated with the data **Clinical Hazards and Harms** The clinical hazards and harms associated with the therapy	**Hazard/Harm Identification** Linking the clinical hazards and harms and assigning probabilities **Phases and Function Identification** Identification of the phases and the associated functions associated with the delivery of a single therapy or exam **Hazardous Situation Development** Identifying the sequence of events and the hazardous situations **Mitigation Development** The development of the essential requirements, the requirements that mitigate risks	**Essential Requirements** The mitigations associated with the inherent risks of the therapy

Risk analysis identifies the essential requirements, that is, the actions or mitigations that ensure the inherent safety of the therapy or procedure. These essential requirements represent key inputs for concept elaboration and the development of the overall design inputs. The workflow for risk analysis follows the workflow noted in ISO 14971 Annex E (ISO, 2012). The following figure details that workflow



NOTE: *P*1 is the probability of a hazardous situation occurring.
*P*2 is the probability of a hazardous situation leading to harm.

Reference ISO 14971:2007

Figure 3-7 Risk Analysis Workflow

The following table details the definitions associated with risk analysis

Table 3-4 Risk Definitions

Item	Definition
Harm	Physical injury or damage to the health of people, or damage to property or the environment. Harm is established by the Medical team and directly relates to the possible consequences of hazards associated with the therapy or procedure

Item	Definition
Severity	The quantification/scaling of the effects of the harm
Hazard	A potential source of harm
Failure Mode	A failure, either of or t the device or the user, that leads to a hazard.
Sequence of Events	The failure, and subsequent actions/activities that lead to a hazardous situation.
Hazardous Situation	Circumstance in which people, property, or the environment are exposed to one or more hazard(s). A hazardous situation is a composite concept, combining a specific hazard, failure mode and sequence of events.
Risk	The probability of occurrence of harm and the consequences of that harm
Residual Risk	The risk remaining after the application risk control measures
Risk Evaluation	The determination of the acceptability of the residual risk

Most previous approaches to implementing the concepts of ISO 14971 Annex E have taken a device centric approach relative to the identification of hazardous situations and harms. But as shown in the following figure, the device centric approach does not address the full range of possible hazardous situations associated with the therapy or procedure

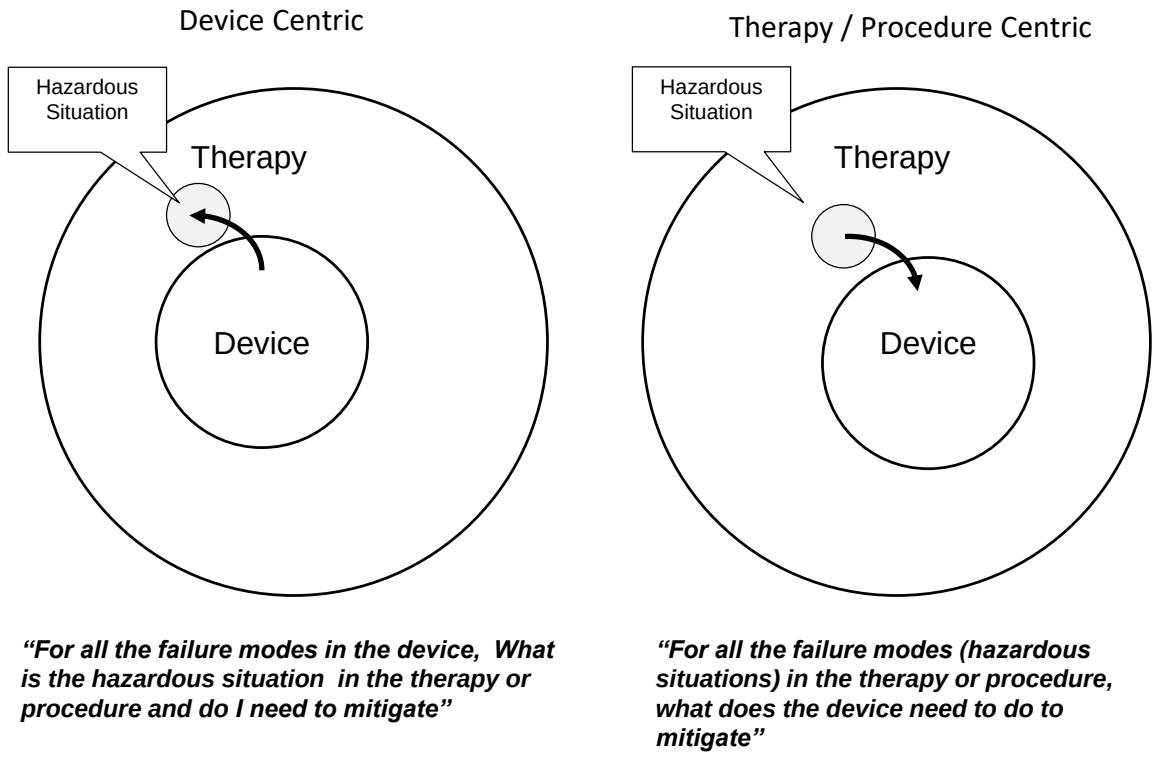


Figure 3-8 Device Centric versus Therapy/Procedure Centric Analysis

The therapy or procedure centric approach aligns well with the safety case approaches put forth by the FDA in recent years. The FDA (Chapman, 2012) defines a safety case as the following

> *A structured argument, supported by a body of evidence, that provides a compelling, comprehensible and valid case that a system is safe for a given application in a given environment*

Safety cases focus on the safety of the therapy or procedure, and how the device performs in supporting the safety of the overall therapy or procedure. In addition, the FDA (FDA, 2014) stress that the safety claims associated with the safety case analysis should focus upon the mitigation of the following types of hazardous situations

- User interface and human factors related
- Software-related
- Electrical
- Mechanical
- Operational
- Environmental,
- Biological

- Chemical,

Clearly, a device centric approach cannot address the full scope of mitigations associated with safety case analysis.

In addition to developing an approach that meets the needs of safety case analysis, the approach to risk analysis requires addressing an increased focus on usability and use errors. Standards such as IEC 62366-1:2015 (IEC, 2015) stress an approach that analyzes the clinical application to identify situations of misuse. These foreseeable misuse situations must be addressed by the design.

The therapy centric risk process detailed in the following sections derives from what has been come to be known as criticality analysis (IEC, 2006). Criticality analysis, in keeping with its linkage to military actions, focuses on steps needed to execute a "mission" and the failures that can impact the execution. As defined in criticality analysis, a failure is linked to the impact upon the overall "mission". The same failure can have different criticality based upon what mission activity or operational phase is associated with the failure. This establishes the following linkage.

Failure → Mission Activity → Mission Impact

In a therapy centric approach to risk analysis, the "mission" is the delivery of a single therapy or execution of a single procedure. Mission impact is the harm to which the patient may be exposed. In the context of ISO 14971 definitions, criticality analysis systematically maps failures to hazards, hazardous situations and the overall therapy. The basic flow follows that of Figure 2-3, specifically

1. Identify the Hazards and Harms for the device
2. Identify all the hazardous situations using the concept of operational phases
3. Assign risk to each hazardous situation based upon the operational phase
4. Identify mitigations when necessary

3.2.1 Hazard/Harm Identification

The therapy centric risk analysis begins with the identification of the Hazards and the associated harms for the therapy or procedure. In the therapy centric analysis, the hazard represents the final element of the causal chain. With this approach, the hazards, a potential source of harm, should be specific to the therapy or procedure. Mapping to the therapy or procedure makes it easier to identify the possible harms associated with the hazard.

For example, the hazards associated with a drug delivery therapy can be characterized as

- Over delivery of drug by X% or over
- Over delivery of drug by less than X%

- Under delivery of drug by Y% or over
- Under delivery of drug by less than Y%
- Delay of therapy by over some time
- Delay of therapy by less than some time
- Interruption of therapy by over some time
- Interruption of therapy by less than some time
- Exposure to particulate matter (non-biologic)
- Exposure to biologics
- Exposure to EMI/EMC
- Exposure to electrical current
- Exposure to hot surface
- Physical impact (drop or otherwise)
- Unintended release of patient information

Additional Hazard Identification guidance can be found in EIN 60601-1 3rd edition in the risk management tables. This list can be modified or extended for the specific medical application, but many of these apply to all devices.

As noted, linking the hazards to the therapy or procedure allows a simpler mapping of harms to each hazard. Harms represent the impact of a hazard, and with the understanding of the therapy, the understanding of the harm is simplified. Each harm has an impact on a patient or clinician, and this impact has a severity. Severity for harms is usually characterized as follows

Table 3-5 Severity of Harm

Severity of Harm	Description
Catastrophic	Results in patient death.
Critical	Results in permanent impairment or life-threatening injury.
Serious	Results in injury or impairment requiring professional medical intervention.
Minor	Results in temporary injury or impairment not requiring professional medical intervention.
`Negligible	Little or no injury

Each specific patient or user harm may have a wide range of impacts, with each impact having a specific severity and probability. As an example, infections due to biologics as a result of a therapy may have little impact (negligible) or may lead to death (catastrophic) in some rare cases. The distribution of impact is a probability distribution as shown in the following table.

Table 3-6 Harm Severity

Harm	**Severity**				
	Catastrophic	**Critical**	**Serious**	**Minor**	**Negligible**
Infection	0.000010	0.00004	0.00005	0.40000	0.59990

The harm severity distribution should be developed based upon historical clinical data and a detailed analysis of the therapy or procedure.

After the identification of clinical hazards and characterization of harms, the harms are mapped to the hazards. The mapping must consider that a single hazard may result multiple harms. As an example, over delivery may result in several harms, with each harm will having an associated distribution of severity. The following table shows the linking of harms severity distributions to a specific hazard.

Table 3-7 Mapping Harms to a Hazard

	Harm Likelihood	**Severity Probability (from the harm analysis)**				
Hazard – Over Delivery		Catastrophic	Critical	Serious	Minor	Negligible
Associated Harm #1	0.5	0.000010	0.00004	0.00005	0.40000	0.59990
Associated Harm #2	0.5	0.000010	0.00004	0.00005	0.30000	0.69990

The weight is the likelihood of the particular harm resulting from the specific hazard. This weight allows the development of an aggregated probability for each severity level. The aggregated probability of each severity can be characterized as shown below.

Table 3-8 Hazard Severity Probability Distribution

	Severity Probability				
Hazard	Catastrophic	Critical	Serious	Minor	Negligible
Over Delivery	0.000010	0.00004	0.00005	0.40000	0.59990

In this case the sum of probabilities across all of the severities is one, that is, the distribution of probability for a given hazard against the severities must sum to 1.

$$\sum_i Probability\ of\ Severity\ level\ i = 1$$

An alternative qualitative approach assigns a single, most likely severity to each hazard. With this qualitative approach, care must be taken assigning severity. The tendency during severity analysis is to assign a severity of "catastrophic" because of some small but finite probability of death exists. This tendency for a qualitative approach to default to a "catastrophic/improbable" severity/probability combination for a given hazard will distort subsequent hazard analysis. Even with a qualitative approach, awareness that each hazard results in a variety of harms and severities is needed to assign the "most likely" severity

The medical team most often performs this activity of identifying hazards and mapping harms, as the linkage and understanding of harms relates to the therapy or procedure, that is, the medical uses. Probabilities of severity should be established with extensive clinical research.

At the completion of this step the Hazard Identification Table, as shown below, can be completed.

Table 3-9 Hazard Identification Parameters

Hazard Identification Element	Description
Hazard	Short description of Hazard.
Associated Harm	Harm which may occur if Hazard is present.
Probability of Harm (P2)	The probability of occurrence of harm given the hazard is present. This may either the most likely value or a distribution as shown in Table 3-8 Hazard Severity Probability Distribution
Severity of Harm	Severity classification of the harm. This may either be a most likely value or a distribution as shown in Table 3-8 Hazard Severity Probability Distribution

3.2.2 Hazardous Situation Development

Identification of all the hazardous situations associated with the therapy or procedure requires the identification of all actions (both user and device) associated with the execution of the

therapy or procedure. Use of functional analysis, rather than starting with the failure modes of the device and assessing risk of the linked therapy or procedure actions, aligns with the current approaches recommended by IEC 62366 and recent FDA publications ((Administration, 2016).

This approach of identifying all actions and the associated hazardous situation has the following advantages

- The risk profile of the device is identified within the context of its use.
- Use Errors, labeling and other failures are a natural result of the approach
- Traditional mitigations are easily identified.

The following is an example of identifying use errors and traditional mitigation

> *An infusion pump warns users to disconnect from the patient prior to purging the delivery path. This is not a failure of the device but rather a response of the device to a potential use error*.

The first process step identifies the overall “mission”. In most cases the “mission” is the delivery of a single therapy or execution of a single procedure. The following drawing (Barba, 2004) is a pictorial of a single “mission”, consisting of the operations associated with a single drug delivery

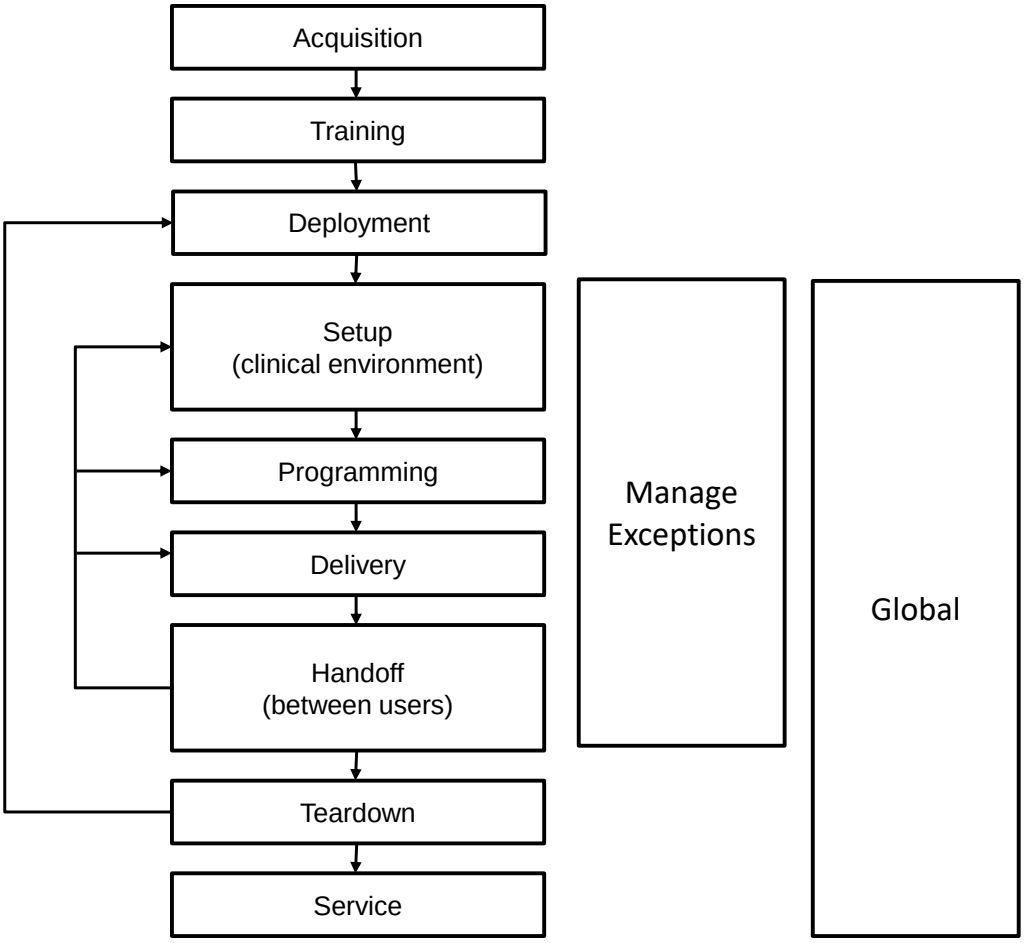


Figure 3-9 Therapy Phases

The following table summarizes each operational phases identified and the possible failure modes

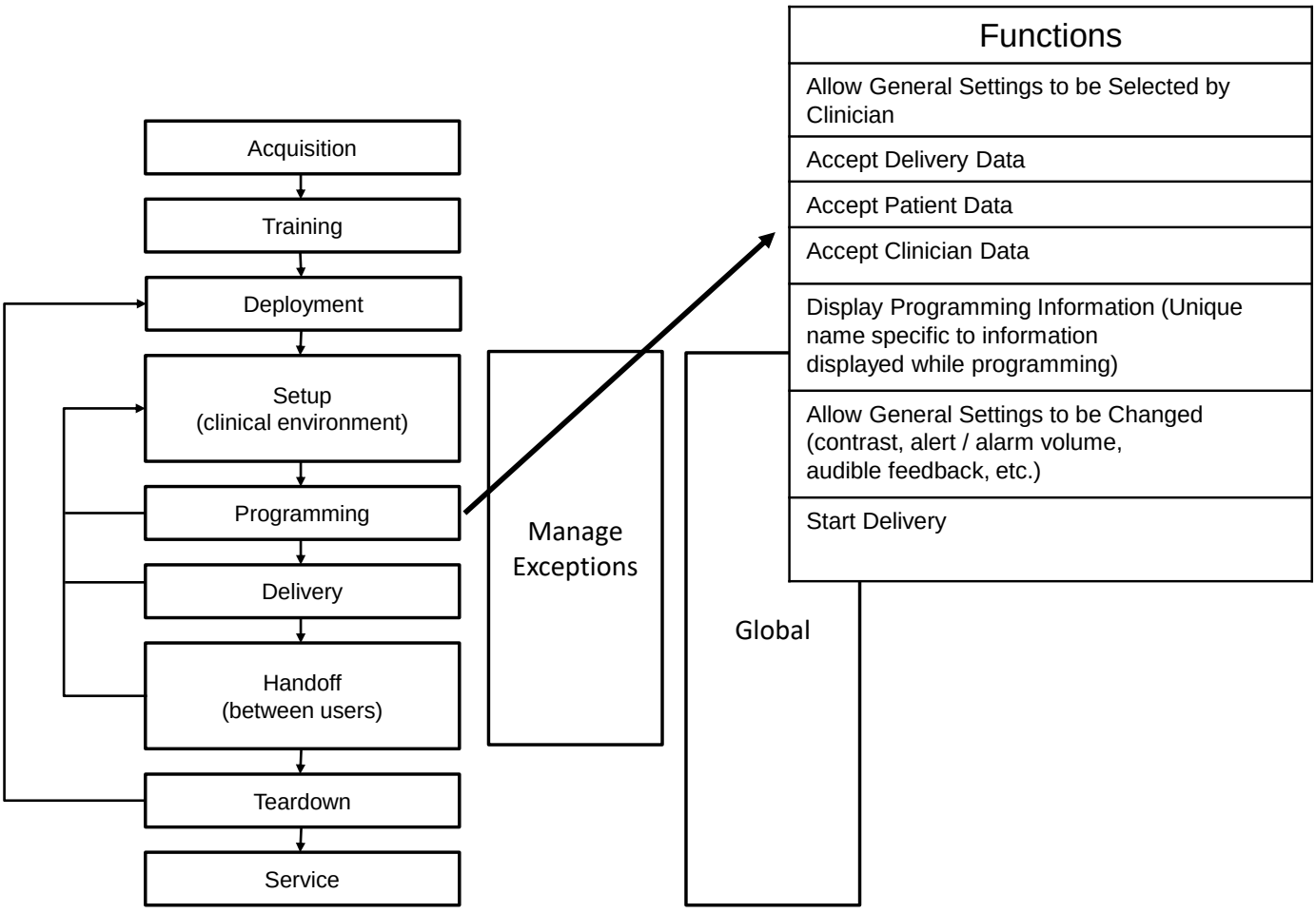
Table 3-10 Therapy Phase Descriptions

Phase	Description
Acquisition	The device ships from the manufacturer to the user. Possible failure modes would include shipping damage
Training	The training of the users Possible failure modes might failure to complete training
Deployment	Moving the device to a position of use, such as loading a disposable into the COW (computer on wheels) Possible failure modes might be storing a prefilled device in the wrong dose bin due to labeling or physical damage while pushing a device on wheels.
Setup	Setting up for delivery of the procedure. Possible failure modes might include improper configuration by the clinician
Programming	Programming the device for the procedure. Possible failure modes might include setting the wrong dose
Delivery	Delivering the therapy. Possible failure modes might include failure of the device to deliver

Phase	Description
Handoff between users	This is the presentation of status during shift changes. Possible failure modes might be failure of the clinicians to discuss status when changing shifts.
Teardown	This is tearing down the setup for the procedure. Possible failure modes might include sharp objects being exposed
Service	This includes cleaning and service. Possible failure modes might include allowing cleaning solutions into the device
Manage Exceptions	Handling alarms and issues. Exceptions are generally handled outside of normal operational context. Possible failure modes might include unnecessary interruption of the therapy
Global	Global includes things such delivering electrical power – those things that move across other phases Possible failure modes might include battery depletion.

Using the operational phase approach identifies the full scope for a therapy delivery or procedure and all possible functions that could be associated with a failure and hazardous situation.

The next step identifies the functions associated with each operational phase. As shown in the following figure, each phase is decomposed into 10-12 constituent functions.



At this point in the process all operational phases and associated functions have been identified.

Following the identification of the functions, a multi-disciplined team should identify the failure modes and sequence of events for each function. A failure for a function is an initiating event that will trigger the hazardous situation. The sequence of events for each failure is the subsequent events leading to a hazard. The division of the failure mode from subsequent events will help drive identification of potential mitigations. The following example describes the differences

- During delivery, over delivery by the device is detected. This is the failure mode
- The sequence of events is the improper clearing of the alarm, resulting in an interruption of delivery

As part of the failure analysis, the operational phase determines the hazard resulting from the failure. As an example, failure of the delivery pump can create an interruption of therapy during the delivery operational phase, but failure will result in delay of therapy hazard should this occur during setup.

The process must accommodate the fact that a single function and the associated failure modes may create may create multiple hazardous situations, depending upon the sequence of events. A single function may be associated with multiple hazardous situations, with each hazardous situation having a unique combination of failure mode, sequence of events and hazard. As a benchmark, each function may have as many as 5-10 distinct hazardous situations. The table below describes what the combination may look like

Table 3-11 Hazardous Situation Combinations

Function	Failure Mode	Sequence of Events	Hazard	Severity
Delivery	Delivery H/W fails	Operator successfully switches to back up system	Interruption of therapy less than 2 min	Minor
Delivery	Delivery H/W fails	Operator fails to switch to back up system	Interruption of therapy greater than 2 min	Critical

The following elements should be identified for each hazardous situation.

Table 3-12 Hazardous Situation Definition

Hazardous Situation Element	Description
Failure Mode	The way in which a particular process input, function, or item being assessed fails or could fail.
Failure Cause	One or more variations in the process that lead to the occurrence of the failure mode.
Sequence of Events	The mechanism that causes a failure mode to become a hazardous situation.
Hazard	The associated Hazard
Hazardous Situation Probability (P1)	Probability of hazardous situation. The probability links to the exposure as shown in Figure 3-7 Risk Analysis Workflow
Harm Probability (P2)	From Hazard Identification Table

Hazardous Situation Element	Description
Severity of Harm	From Hazard Identification Table
Risk Probability (P1xP2)	Combined Probability of Occurrence for the hazardous situation.
Mitigation or Control (Critical Requirement)	These are defined in the following section
Disposition	Disposition on whether the field performance meets target risk.

The determination of the probability of exposure (P_1) is the next step in the analysis of the hazardous situation. Hazardous situation probabilities are based upon occurrence per therapy or procedure. Calculations based upon hours and other measures should be avoided, as these measures tend to distort the real failure rates. Most regulatory agencies have settled on failure rate per therapy or procedure as the preferred rate.

The breakdown of the overall probability of a Hazardous Situation (exposure) *i*, $P1_i$ is given as

$$P1_i = \lambda_i * \alpha_i * B_i$$

Where

λ_i – The failure mode associated with the particular exposure (hazardous situation)
α_i – The normalization factor for the particular mission phase associated with the failure
B_i – The conditional probability $P(HazSit\ i \mid \lambda_i)$ will occur, given the failure mode

The constraint on α_i is that $\sum_i^{all\ mission\ phases} \propto_i = 1$

Note that these parameters are defined in IEC 68012, section 5.3.4 (IEC, 2006).

As an example, for a keyboard failure (the failure mode) with the delivery device, if 90% of the keyboard use occurs during programming the delivery, the α_i associated with λ_i of a keyboard failure during the programming of the delivery resulting in over delivery would be given as

$$P1(Over\ Delivery\ during\ Programming\ due\ to\ keyboard\ failure) = (1.0 \times 10^{-5}) * .9 * (1.0 \times 10^{-3})$$

$\lambda_i - 1.0 \times 10^{-5}$ is the rate of keyboard failure (1 in every 10^{-5} overall therapies)

α_i – 90%, the relative time that typing occurs during programming

B_i – 1.0×10^{-3} is the probability that Over Delivery will occur given a keyboard failure

In reviewing this example, the keyboard failure is the overall rate of a keyboard failure, with 90% (α) of failures happening in the programming phase (logically, this is where most typing occurs). Because users generally check the typing and they system includes other checks, ß indicates that only 1 in every 1000 keyboard failures actually lead to programming over delivery.

The next step involves the determination of P2. The determination of P2 connects the hazard for the hazardous situation to the harm. During this process, it should be noted that the probability of a particular severity of harm for a hazard is not automatically 1. As an example, only about 1 in 1000 interruptions of therapy may result in a critical severity harm, with the majority of these hazards resulting in a severity of minor or negligible. Each hazard has a probability of creating harms with all possible severities (catastrophic to negligible). This can be expressed as

$$\sum_{j}^{all\ j} P(severity\ j \mid hazard\ k) = 1$$

In most cases, an evaluation of severity and probability for all possible severities assigns the severity for the hazard associated with the hazardous situation. This yields the P2 for the hazardous situation, which is the P2 for the associated hazard. Following the determination of P1, the probability of an exposure (hazardous situation), the total probability (P1xP2) of the risk is usually determined using the following table

Table 3-13 Combined P1xP2 Probability

	Probability of Level of Harm given Hazardous Situation (P2)				
Probability of the Hazardous Situation (P1)	Frequent	Probable	Occasional	Remote	Improbable
Frequent	Frequent	Probable	Occasional	Remote	Improbable
Probable	Probable	Remote	Improbable	Improbable	Improbable
Occasional	Occasional	Improbable	Improbable	Improbable	Improbable
Remote	Remote	Improbable	Improbable	Improbable	Improbable
Improbable	Improbable	Improbable	Improbable	Improbable	Improbable

At the completion of the hazardous situation analysis, the combined probability (P1xP2) and associated severity of each identified hazardous situation has been determined.

3.2.3 Hazardous Situation Risk Evaluation

Risk evaluation reviews the probability and severity for each hazardous situation against predetermined criteria. Normally a table as shown below establishes the acceptability of the residual risk for a hazardous situation.

Table 3-14 Risk Evaluation Matrix

	Severity of Harm				
Probability of Occurrence of Harm (P1xP2)	Negligible	Minor	Serious	Critical	Catastrophic
Frequent	Unacceptable	Unacceptable	Unacceptable	Unacceptable	Unacceptable
Probable	Unacceptable	Unacceptable	Unacceptable	Unacceptable	Unacceptable
Occasional	Acceptable	Acceptable	Conditionally Acceptable	Conditionally Acceptable	Conditionally Acceptable
Remote	Acceptable	Acceptable	Acceptable	Conditionally Acceptable	Conditionally Acceptable
Improbable	Acceptable	Acceptable	Acceptable	Acceptable	Acceptable

The ranged Unacceptable, Conditionally Acceptable and Acceptable come from the standards such as ISO 14971:2012 (ISO, 2012). Conditionally acceptable usually requires further analysis and documentation to establish the acceptability. This table forms the basis for the overall acceptability of the risk for the device as well as need for risk controls.

3.2.4 Risk Mitigation and Controls

Based upon the evaluation of risk, mitigation of the hazardous situations will be needed. In determining the need for risk controls, care must be taken to align with the requirements of ISO 14971:2012 (ISO, 2012). The approach to risk control should be based upon the risk for a hazardous situation and can be summarized as follows

Table 3-15 Risk Acceptability Approach

Risk Characterization	Approach
Acceptable	• Are existing risk controls possible? • Will additional risk controls significantly reduce risk?

Conditionally Acceptable	• Do existing risk controls address the concept of "safe by design"? • Can additional controls reduce the risk to an acceptable level?
Unacceptable	• Do existing risk controls address the concept of "safe by design"? • Can additional risks reduce the risk to an acceptable level? • Is the risk inherent in the therapy? What is the clinical risk/benefit?

This approach drives to the application of risk controls in most cases, with clear direction of the approach. In most systems, any "Conditionally Acceptable" risk should have a "safe by design" risk control. "Unacceptable" risks should exist only when these risks are directly related to the therapy itself.

Developing risk controls that implement "safe by design" involve integrating mitigations into the design. The development of design mitigations relies on the analysis of the failure modes and the sequences of events. There are three types of mitigation

- Mitigating the failure mode (λ_i)
- Mitigating the sequence of events (B_i)
- Transform the hazard and lower the severity

Mitigating the failure mode focuses upon improving reliability. As an example, improved components can improve the overall reliability of a delivery pump, reducing the probability of a failure during delivery. This type of mitigation can be an effective mitigation, but is also the most expensive. In addition, the α factor can lower the overall effectiveness in cases where the particular phase does not occupy a significant portion of the overall therapy or procedure time.

Mitigation of the sequence of events involves mitigating the transition from the failure mode to the hazard. This mitigation of the sequence of events represents the key mitigation for use errors involving the device and should be done in conjunction with the identification of the use error type. The following figure (Kaye, 2010) illustrates the taxonomy of use errors

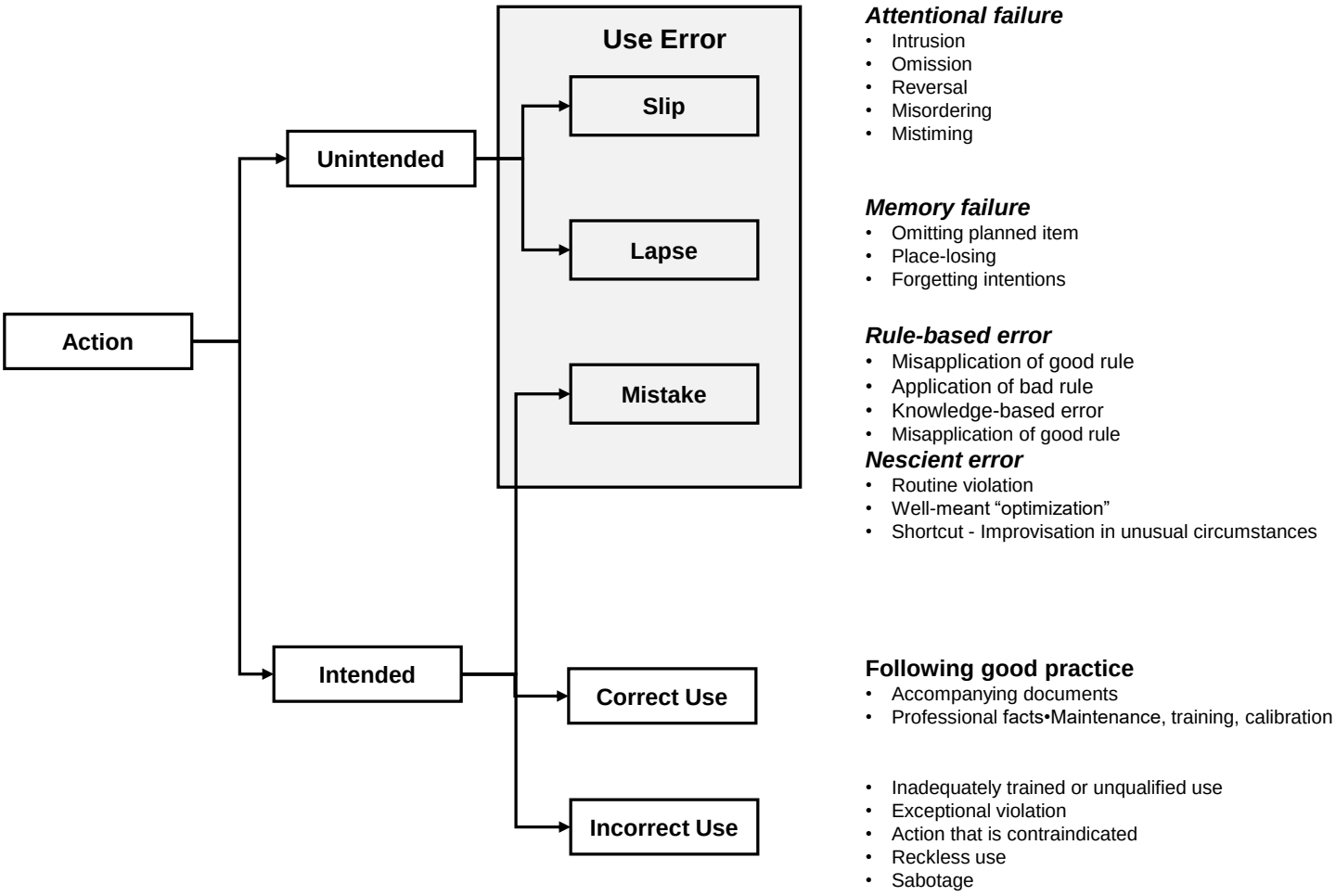


Figure 3-10 Use Error Taxonomy

The characterization of the use error drives the identification of the appropriate mitigation. The following table identifies several established mitigations

Table 3-16 Use Error Mitigations

Use Error Type	Example	Possible Mitigations
Slip	• Failing to identify an error condition • Performing steps in the wrong order when loading a device	• Alarms • Wizards
Lapse	• Forgetting to check dose settings • Failing to purge delivery circuit	• Confirmation prompts • Embedded State machines for device execution

Use Error Type	Example	Possible Mitigations
Mistake	• Assuming the Device is working properly • Skipping steps to “optimize” the execution	• Confirmation prompts • Wizards

The process of identifying the correct mitigation for a use error may often involve observational studies during the risk analysis. Without knowledge of how users will respond to the proposed mitigations, the effectiveness may be limited. In addition, failure to identify the correct mitigation can have a significant impact on validation, where the effectiveness of the mitigation must be confirmed.

In addition to addressing user error, mitigating the sequence of events can be used to mitigate hardware or systemic issues. As an example, keyboard entry failures (mistyping) occur with a very high frequency (sometimes as much as one in every 20 entries). The use of a confirm screen does not eliminate the failure mode, but does mitigate the sequence of events (the mistyped entry is accepted and programmed into the device). Usually this type of mitigation is not considered as effective as mitigating the failure mode.

Transformation of the hazard associated with a hazardous situation is best illustrated through the application of Power-On-Self-Test (POST) as a mitigation. In this situation, the hazard is transformed from interruption to delay, with a corresponding reduction in the severity of associated harms. This is generally considered the most effective mitigation, as lowering the severity quickly moves the risk to acceptable. This type of mitigation often links closely to the operational and functional design and need to be identified and developed early in the development cycle.

Mitigations will form the essential requirements developed as part of the risk analysis process. In many cases a similar mitigation will apply to many hazardous situations and the final step is to collate the list of mitigations into a single set of requirements.

3.2.5 System Risk Evaluation

Overall system risk can be evaluated heuristically or mathematically. In heuristic evaluation, criteria such as

- No unacceptable hazardous situations
- Conditionally acceptable less than 20% of all hazardous situation

Become the basis for accepting the system risk and the acceptability of the mitigations as requirements.

A more mathematically based approach requires significantly effort. An example of a mathematical criteria may be as follows

The total probability of critical severity hazardous situations should be less than 1 in every 10000 ($1x10^{-4}$) therapies.

Calculating the total residual risk (probability by severity) starts with the mapping of the hazard across the full range of severities. Each hazardous situation (exposure) event $P1_i$ will result in a specific $hazard\ k$, and each $hazard\ k$ could result in all possible severities of harm, that is

$$\sum_{j}^{all\ j} P(severity\ j \mid hazard\ k) = 1$$

The severity probability $P(severity\ j \mid hazard\ k)$ is determined from the

$$\sum_{i}^{All\ Harms} \mu_i \sum_{j}^{all\ j} P(severity\ j \mid harm\ i) = 1$$

Where μ_i is the normalization factor for the harm, given the specific hazard. This can be expressed

$$\sum_{i}^{all\ harms\ for\ a\ given\ hazard\ k} \mu_i = 1$$

The following mathematical evaluation calculates the total residual risk. The total residual risk of a given $severity\ j$ is given as

$$P_{total}(severity\ j) = \sum_{k}^{all\ hazards} P2(severity\ j \mid Hazard\ k) * \sum_{i}^{where\ situation\ hazard\ is\ k} P1(exposure\ results\ in\ hazard\ k)_i$$

By summing over all hazards and all hazardous situations with the resulting hazard we can compute the total probability of the device use resulting in an outcome of a given severity.

3.3 Elaborating the Concept

Elaborating the concept integrates the results of the concept definition into a complete system definition, the design inputs.

Table 3-17 Elaborating the Concept SIPOC

Inputs	Key Activities	Outputs
Clinical Use Clear understanding of the overall therapy, workflow and risks associated with the seminal idea **Risk Analysis and Essential Requirements** The phase, function and mitigations developed as part of the risk analysis step. **CTQs** The key requirements identified as part of the concept selection process **VOB – Business Requirements** The Needs of the business **Standards** Regulatory as well as business standards consistent with the markets for the product **Manufacturing Capability** Characterization of capacity, measurement systems and process capability	**Use Case Development** a structured analysis of the clinical application to develop system modes of operations and interactions with other elements **Architecture Elaboration** Breaking the system down to into subsystems **Requirements Elaboration** Translation of the Performance Parameters, CTQs and concept into the system definition (use cases, requirements, architecture and interfaces) **Final Concept Confirmation** The final confirmation of the concept with the users	System Definition (Design Inputs) • System and Subsystem Requirements • Use Cases, Risk Control • Preliminary Architecture and Interfaces

The overall work flow is as shown in the following

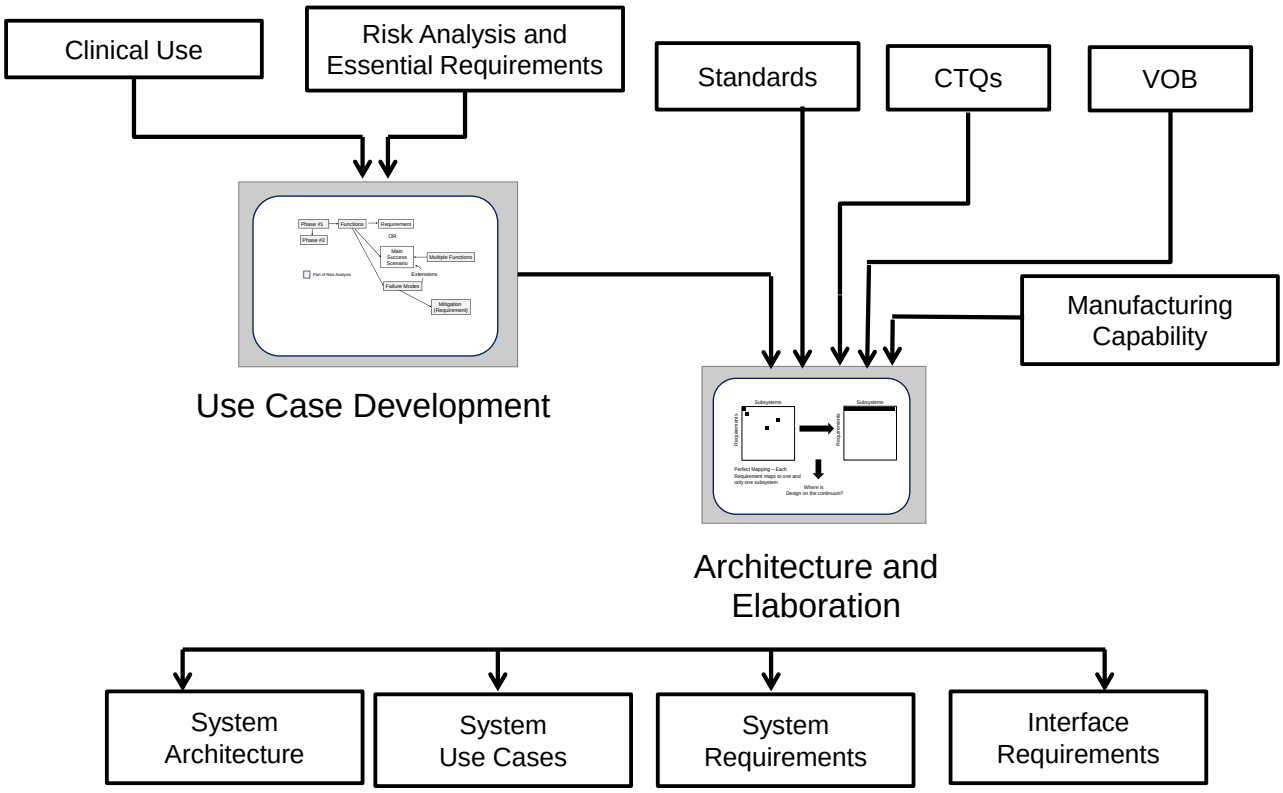


Figure 3-11 Elaborating the Concept

3.3.1 Developing Use Cases

The operational risk analysis (section **Error! Reference source not found.**, **Error! Reference source not found.**) forms the basis for the use case development. During the operational and functional analysis portion of the risk analysis, use cases have been identified and linked to the failure modes. This analysis needs to be formalized into use cases that can be translated into requirements. In most definitions of a use case (Cockburn, 1998), the following elements are initially identified

Table 3-18 Use Case Top Level Elements

Use Case Element	Description
Use Case Name	Descriptive name for the use case
Scope	What is the scope of the use case. This should relate back to the mission or therapy phase.
Level	This should be either Summary, Primary task (phase), Sub-function (a sub-case within the primary task)

Use Case Element	Description
Priority	Within this context this should be related to the maximum severity of the associated risk
Frequency	This should be linked to the α as defined in the risk analysis
Trigger	What initiates the use case
Main Success Scenario	What is the main path for the use case, often referred to as the “happy path” in that errors and issues are not addressed
Extensions	Additions to the use case, usually in the case of issues or failure modes.
Sub-Variations	Explicit differentiated changes to the use case

The operational phase risk analysis links failures to use cases. The following figure illustrates how the elements of the risk analysis integrate with the workflow in the development of use cases

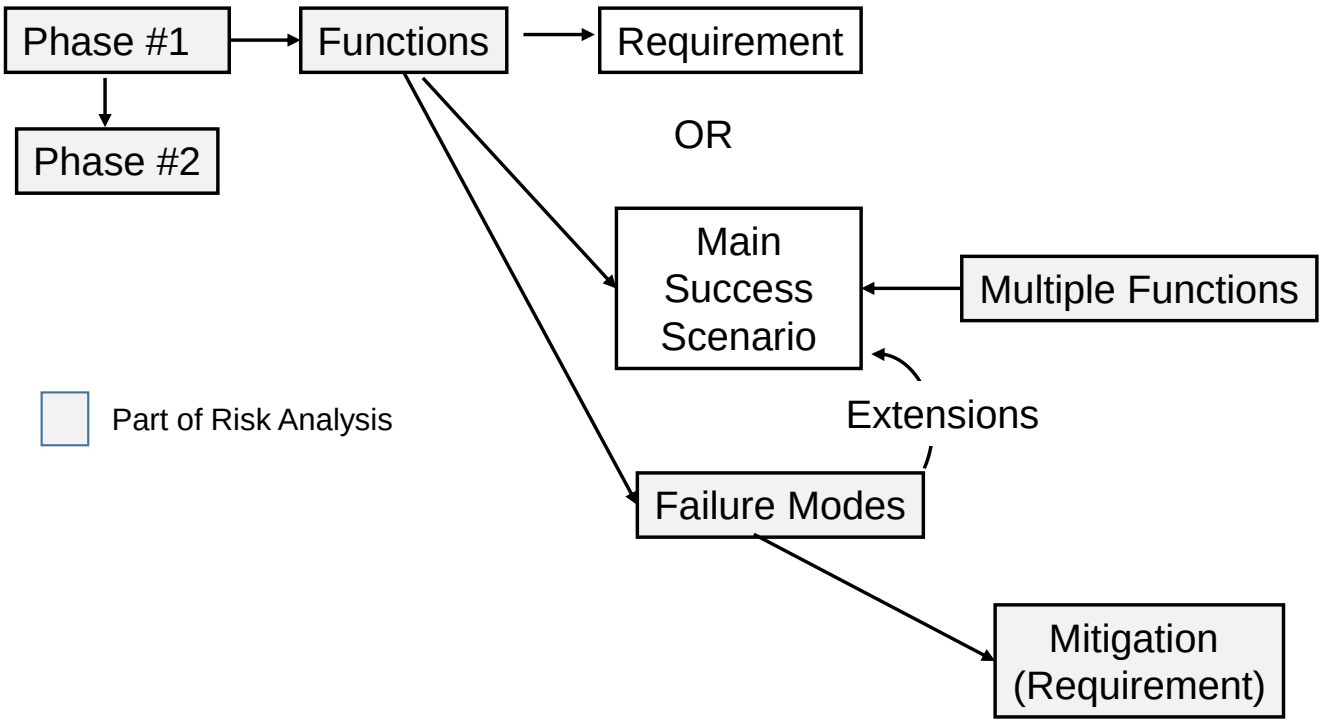


Figure 3-12 Use Case and Risk Analysis Linkages

The development of the use cases should consider the following

- **Completeness** - Main Success Scenarios may link to one or multiple functions, as identified in the risk analysis, depending upon the granularity of the use cases

development. Simpler systems may require fewer use cases, but all Main Success Scenarios must address all phases and associated functions.

- **Extensions** – Each Use Case must include extensions for all of the failure modes associated with the functions included.
- **Linkage** – Each use case must include information that links the Use Case to the mitigations associated with the failure modes encompassed in the extensions.

3.3.2 Architecture and Requirements Elaboration

The development of the final set of requirements closely couples to the development of the architecture. Architecture embodies the breakdown of the system into subsystems, and the final system requirements development should consider the architecture. Many approaches to medical device development advocate for design inputs that are "concept free", but a concept free approach can lead issues when the design inputs are translated to lower level requirements. Specific areas that can be addressed through consideration of the architecture of the system include

- **Subsystem Development and Integration** – a system where requirements map more directly to the subsystems lessens the pressure to ensure that subsystems and interfaces are consistent during development. Subsystem development can proceed without the necessity to review and evaluate design through the development cycle.
- **Subsystem Verification** – Verification at the subsystem level can satisfy the overall system verification, usually with much less cost. As an example, a single drug delivery subsystem with that delivers all of the critical requirements for accuracy may be tested using "white box" methods that are much more efficient and accurate.

Overall, the careful consideration of the structure of the requirements and the associated architecture results in a system more amenable to parallel subsystem development, cleaner integration and simpler verification.

3.3.2.1 Architecture Evaluation Criteria

The elaboration process begins with identifying the goals for the subsystem decomposition. Manufacturing capability, the development approach (insource/outsource) and organizational factors drive the development of goals for the subsystem decomposition. Key business elements for consideration include

- **Voice of the Business** – in some texts this is referred to as "non-functional requirements", but represents the product from an institutional view
- **Manufacturing Capability** – this is the "how" for the production, and needs to consider make/buy and other critical factors.
- **Pre-existing Systems** – in almost all system development, pre-existing systems or components exist

In addition to the business considerations, design considerations should be established. These should include

- **Coupling** – the degree to which subsystems depend upon each other
- **Cohesion** – the degree to which the elements of a subsystem functionally belong together.
- **Subsystem Structure** – This should consider organizational structure and the alignment of the architecture with the structure of the development group. A old saying, "the system architecture reflects the organization" should not be ignored.

The elements should be formed into a set of "needs" and given the familiar 1/3/9 weighting. Only one or two of the design or business needs should be assigned a 9 weighting. The following figure shows the Pugh Matrix for architecture

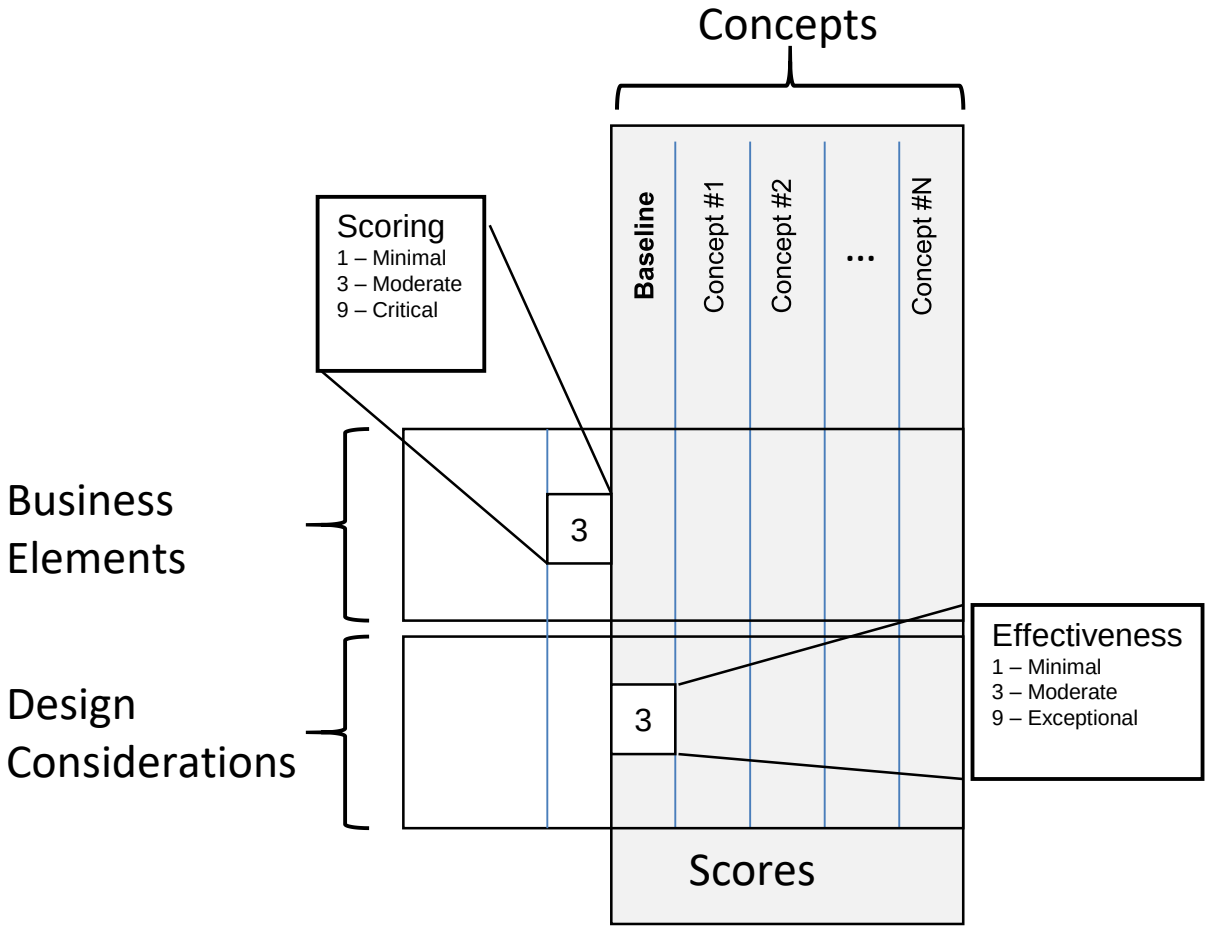


Figure 3-13 Requirements Elaboration Pugh Matrix

The process of developing the system architecture relies on the same iterative techniques used in the development of the concept. At least two passes, with the development of a super concept, should be performed. This will drive the development and evaluation of

alternatives for the architecture and prevent rushing to an unsuitable partitioning with issues that will only surface during the subsequent stages of development.

Definition of the architecture is best performed prior to the finalization of requirements. As will become apparent, tailoring the final system requirements to the architecture will result in a system that is easier to integrate and verify. The following illustrates the general flow

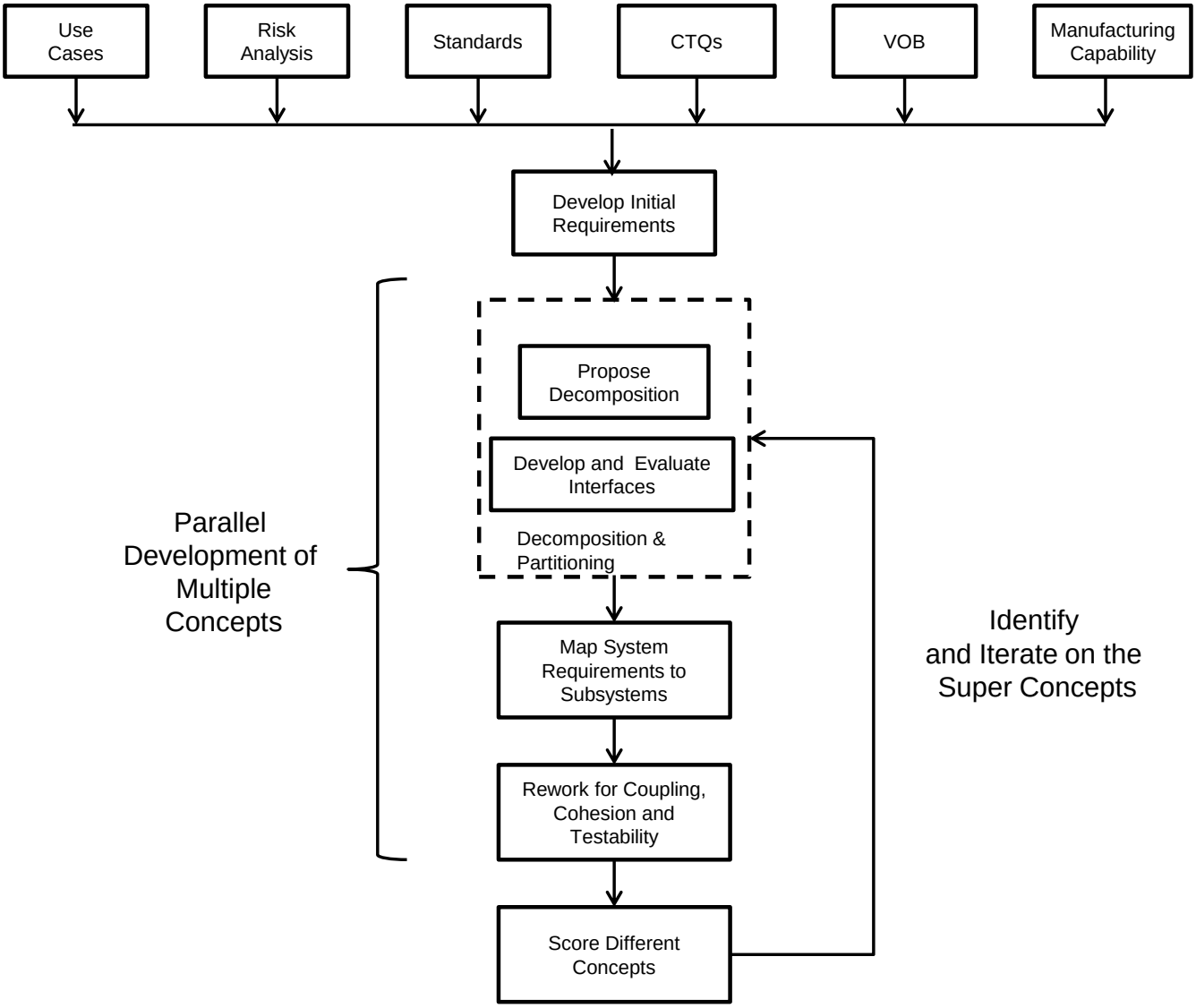


Figure 3-14 Requirements Elaboration Workflow

This workflow replicates the workflow used in concept development. Wherever possible several sub-teams should develop proposed architectures. The whole development team evaluates the proposals and develops the super concept for the iteration. This should continue until the team decides that further iteration will not substantively change the outcome.

3.3.3 Subsystem Decomposition

Decomposition partitions the system into subsystems. Rarely in today's world does a system of any complexity not have a subsystem-based architecture. Subsystems satisfy several key needs of modern medical devices

- **Parallel Development** – with lower coupling comes fewer interdependencies and thus more parallel effort can be planned.
- **Outsourcing** – Subsystem partitioning allow the outsourcing of functionality to groups or organizations with specialized skills
- **Fabrication and Manufacturing** – manufacturing almost always starts with the build of subcomponents. If these subcomponents match the subsystems, the development of test methods for manufacturing can be significant lowered
- **Post Launch Changes** – following release, subsystems allow more structured test and introduction of design changes

As part of the decomposition activity, the decomposing the system to subsystems represents a significant part of the effort. At the system level, the characteristics of the subsystems and interfaces become an important design consideration. Key interface properties are as follows

- **Service Granularity** – the level of interaction between subsystems. Usually this is defined as either a *coarse* or *fine*-grained interface granularity.
- **Managed Information** – the data structures that are managed by the subsystem and the external visibility of these data structures.
- **In-Band versus Out-of-Band Control** - the degree to which the interface to a subsystem supports asynchronous out-of-band interactions

The design principle of Service Granularity identifies the optimal scope subsystem interactions. A coarse-grained service operation has broader scope than a fine-grained service, although the terms are relative. The former typically requires increased design complexity but can reduce the number of calls required to complete a task. The four key factors to consider when designing for optimal granularity are performance, message size, transaction and business functionality.

As an example, the interface to a drug delivery pump subsystem may consist of a single, all-encompassing delivery command, with all the delivery parameters, or it may consist of several sequenced commands, where the controller of directly sequences the delivery. The structure chosen will depend upon system considerations, but the direct sequencing will have more granularity and high coupling, which generally places more dependencies on development and testing.

Managed Information refers to the amount of information a subsystem makes available to the rest of the system. In object oriented programming, the data within an object may have a scope of private or public. Generally, public data is kept to a minimum, lowering the coupling between objects. In subsystem decomposition, the minimization and isolation of data signals

should be carefully considered to lower the subsystem coupling.

As an example, electrical signals from a pressure sensor used only as part of the delivery function might best be maintained within the delivery subsystem.

Managed Information is a key element of the SNMP (Simple Network Management Protocol), with the ASN.1 protocol defining the structure of the (ITU-T, 2008). This represents a structured way of presenting data between connected systems (subsystems). While this may be overkill for the definition of subsystems, it nonetheless represents a real-world model that will drive better definitions of the managed information across a system.

Defining the managed information requires identifying the following elements of the data

- **The data hierarchy and organization** – what data elements are part of what groups. Data elements within a group should usually read or set as a group. A proper hierarchy feeds from and supports the service granularity. The use of database normalization techniques (Microsoft) represents a great way to evaluate data organization and hierarchy.
- **The data types** – the type of data (enumeration, integer or floating point number) for data elements. The definition of the data type impacts both the “setting” and “getting” of the data. As a rule, unnecessary precision in a data type represents unnecessary coupling between subsystems.
- **The data visibility** – whether the data is available internally within the subsystem or to connected subsystems significantly impacts overall coupling. “Less is more” is the best approach with visibility.

In-Band versus Out-of-Band control analysis focuses on the degree of asynchronous behavior that the system will support. Out-of-Band control can best be compared to the familiar “interrupt” in computers. With Out-of-Band control, the subsystem or component asynchronously notifies the system of an event and the system schedules the servicing of the event. Alternative approaches using In-Band control use timing loops that regularly poll the various components or subsystems. The choice of In-Band versus Out-of-Band control trades off more subsystem coupling (In-Band) for more complex subsystem design (Out-of-Band).

In analyzing the use of In-Band versus Out-of-Band control, key considerations should be the development of preliminary control flows, the analysis of events and the impact of the event on the control flow. If an event impacts subsequent control operations, Out-of-Band control may be required. As an example, failure of a sensor may require interrupt driven behavior to prevent a control loop from adding invalid data elements to the control loop filter history.

The analysis and definition of the interface characteristics as outlined here drives the scoring of the coupling and cohesion of the system. As architecture evaluation progresses, minimization of the coupling becomes an important measure of the suitability of the architecture. Review of the interface structures during evaluation ensures that the system will have the required performance and coupling characteristics. Once the architecture has been identified, requirements can be mapped to subsystems.

3.3.4 Subsystem Coupling and Cohesion

Considering of the interactions of the subsystem also drives subsystem decomposition. System requirements that depend upon multiple subsystems for fulfillment are not as desirable as those that can be satisfied by a single subsystem.

As part of the decomposition process, a preliminary set of system requirements should be put in place, based upon the CTQs, VOB and risk analysis. Following the decomposition to subsystems, these requirements can be evaluated as for interaction against the subsystems. The following shows the structure of this evaluation

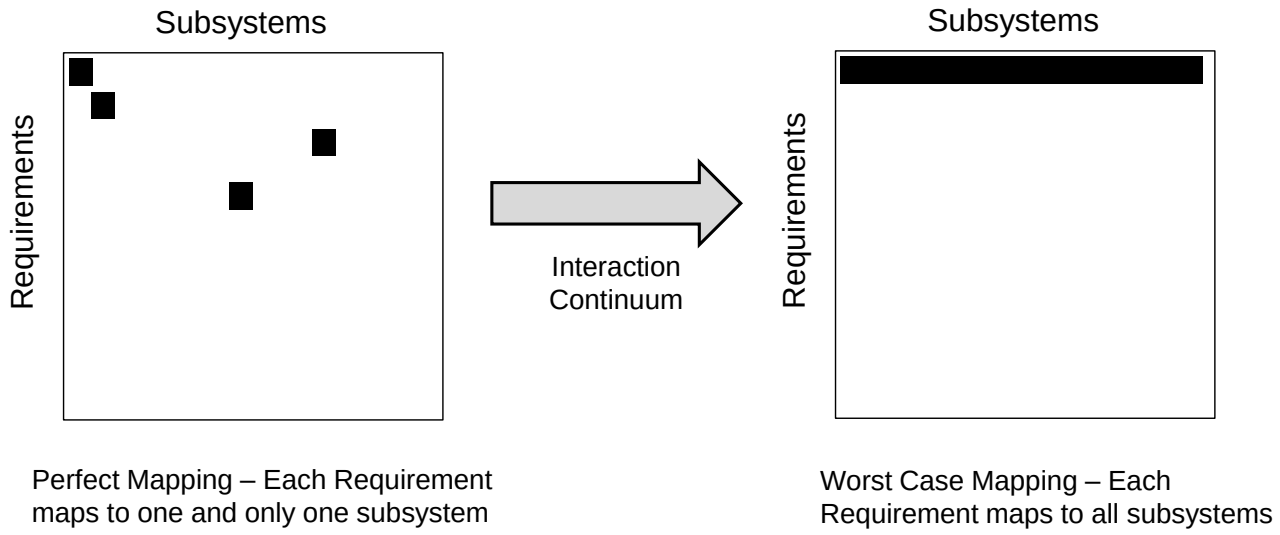


Figure 3-15 Requirements to Subsystem Mapping

This evaluation strives to establish a subsystem breakdown with minimal interaction. A perfect mapping results in a system with far too many subsystems, whereas a system with a high degree of interaction between subsystems impacts integration and verification.

In addition to the interaction, the subsystem mapping needs to consider the subsystem cohesion. Cohesion is the evaluation of the degree to which functionality within the subsystem belongs together. Interaction mapping drives qualitative evaluation of the subsystem decomposition, as shown below

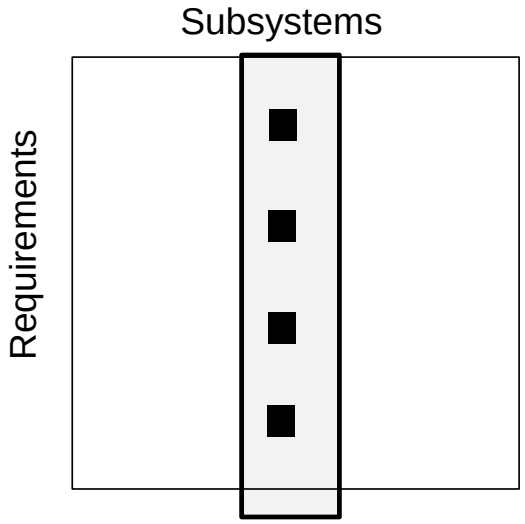


Figure 3-16 Subsystem Cohesion

Iterating on the interactions between subsystems and evaluating the cohesion of the resulting mapping should be coupled with the Pugh Matrix and the super concept process. The expectation is that multiple iterations will be required to get the right architecture.

While iterating on the architecture, iteration on the requirements should also take place. The preliminary set of requirements may be subdivided to yield better mapping and cohesion. Often the initial wording of a requirement results in unwanted interaction. By analyzing the cohesion, system and subsystem requirements may be reworded to improve the cohesion. Cohesion with respect to requirements addresses the question

"Do all the statements or needs in the requirements belong together?"

At the end of the process, subsystem decomposition results in requirements that are consistent, and the desired coupling and cohesion has been achieved. The data and analysis from this step will be used to drive further subsystem requirements definitions later in the process.

3.3.5 Structure of a Requirement

The NASA Systems Engineering Handbook (NASA, 2007) defines the following guidelines for requirements

- The requirement is in the form "product ABC shall XYZ." A requirement must state "The product shall" (do, perform, provide, weigh, or other verb) followed by a description of what must be done.
- The requirement uses consistent terminology to refer to the product and its lower level entities.

- Complete with tolerances for qualitative/performance values (e.g., less than, greater than or equal to, plus or minus, 3 sigma root sum squares).
- Is the requirement free of implementation? (Requirements should state WHAT is needed, NOT HOW to provide it; i.e., state the problem not the solution. Ask, “Why do you need the requirement?” The answer may point to the real requirement.)
- Free of descriptions of operations? (Is this a need the product must satisfy or an activity involving the product? Sentences like “The operator shall…” are almost always operational statements not requirements.)

When reviewing design and development input documents (this often happens in conjunction with the customer), the following criteria should be checked (see (ISO/IEC, 2014)

- Ambiguities and contradictions
- Inconsistent, incomplete or unfeasible information or requirements,
- Unrealistic performance specifications,
- Requirements that cannot be verified or validated,
- Unstated or assumed requirements,
- Inaccurate description of user environment and actions

3.3.6 Verification Check of the Requirements

When requirements have been developed, a key check is to review the verification of the requirement. The general categories of verification are as follows (Department of Defense, 1994)

- Demonstration (D) - The verification on observable functional operation not requiring the use of instrumentation, special test equipment, or subsequent analysis. As an example, the requirement that “all user input shall require separate and explicit user confirmation” can be verified by demonstrating that all input has a confirm screen
- Test (T) – verification using instrumentation or other special test equipment to collect data for later analysis. As an example, test will be required to confirm the accuracy of delivery is +/- 10%
- Analysis (A) - The processing of accumulated data obtained from other qualification methods. Examples are reduction, interpretation, or extrapolation of test results.
- Inspection(I) - The visual examination of design outputs
- Special qualification methods (S) - Any special verification methods, such as special tools, techniques, procedures, facilities, and acceptance limits.

Characterization of verification should be performed during the development of the requirements and analyzed. Requirements that require several types of verification may be poorly written and should be considered suspect.

3.4 Final Concept Confirmation

Based upon the breakdown of the subsystems and the adjustment of the requirements, the concept prototype as realized following concept selection is updated. Drawings, models or other tangible artifacts are updated and feedback from customers gathered. At this point in the process the concept prototype should have much more fidelity and should be that much closer to the final realization.

The updated concept is again presented to the customers and assessment against the same user needs performed. The method for assessment should the same as the method used in to section 3.1.6, Early Concept Confirmation. This confirmation process confirms that the system, as reflected by the requirements, still meets the needs of users directly affiliated with the entities that purchase the product

4 Developing the Product

4.1 Requirements Deployment

The development the product begins with the deployment of the design inputs to the subsystems. The following table describes the inputs and outputs of Deployment Process

Table 4-1 Requirements Deployment SIPOC

Inputs	Key Activities	Outputs
Use Cases Workflows, both primary and secondary **System Architecture** The subsystem partitioning as defined in the system design inputs. This should include the preliminary mapping used during the concept elaboration **System Requirements** The requirements from concept elaboration, including the essential requirements from the risk analysis **Interface Requirements** Details of the internal and external requirements **Manufacturing Capability** Characterization of capacity, measurement systems and process capability	**Assessing Process Capability** The characterization the measurement system (gage R&R) against Performance Requirements and CTQs, focused on understanding the ability to meet the requirements **Design Worksheet Development** The analysis of the requirements, identifying those requirements that require a transfer function **Transfer System Deployment** Develop transfer functions from results of the Design worksheet development	**System Transfer Functions** For those performance requirements with complex relationships, the details of the transfer function. **Workflow Diagrams** For those functional requirements with complex relationships, the details of the workflow and subsystem interactions. **Design Worksheet** The details of how the system design inputs and subsystem requirements interrelate S**ubsystem Requirements** The subsystem requirements **N^2 Chart** The interactions between the subsystems

A capability analysis of the manufacturing process and the ability of manufacturing to meet the various performance requirements initiates the process. This analysis drives how the subsystem deployment proceeds and may lead to revision of the requirements.

Following confirmation that the manufacturing process can deliver on the requirements, the process focuses on how the requirements distribute between subsystems. During distribution of requirements, design worksheets and transfer functions are developed in places where the deployment of the requirements creates interactions between the subsystems. This may be formulas, design documents or, for functional requirements, workflow diagrams detailing the interactions between the subsystems

During deployment, it may be necessary to revisit and update design inputs, and deployment ensures that this happens well before the realization. This lowers the cost and the associated rework of the process.

At the end of the process, the design worksheet, transfer functions and workflows serve as inputs to the final transfer function deployment. Here the subsystem requirements are reconciled across the design worksheet and the final allocation of tolerances across the subsystems locked down. The transfer functions, subsystem requirements and design worksheet form the basis for the system realization.

The deployment workflow is shown in the following figure

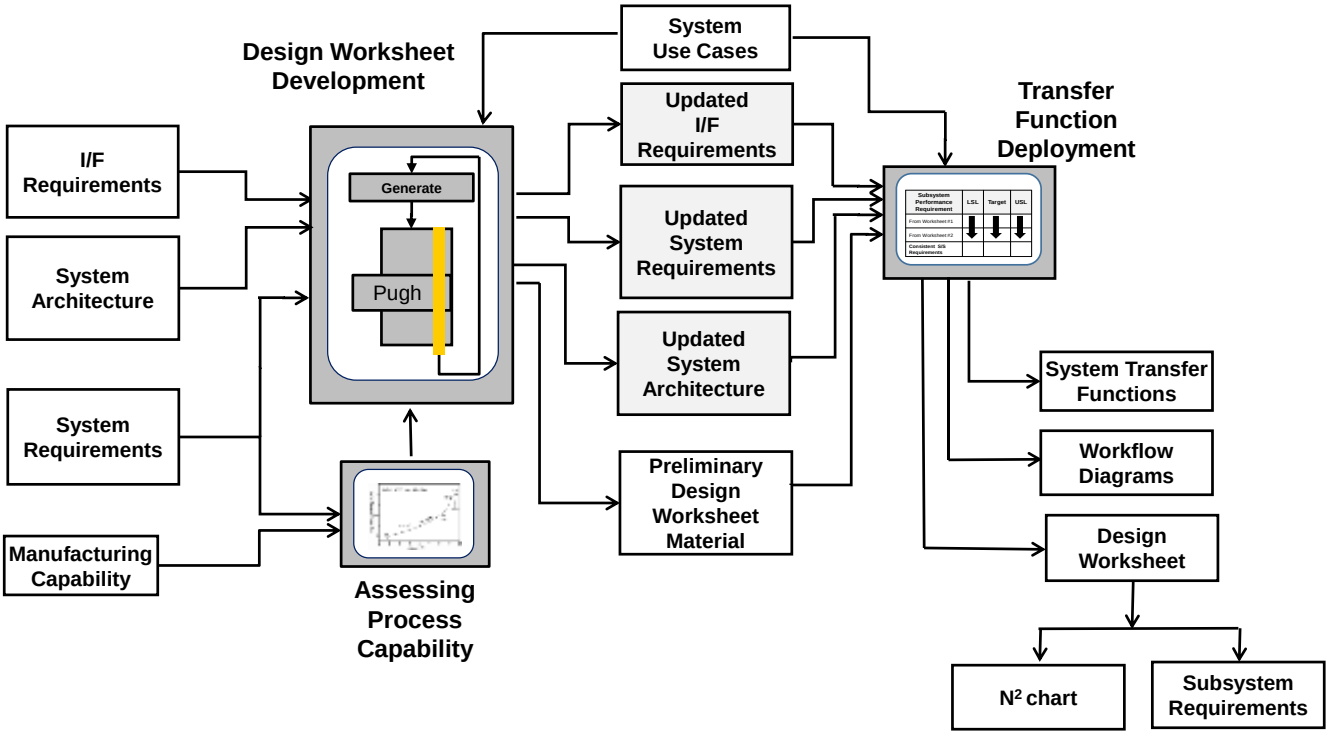


Figure 4-1 Requirements Deployment

4.1.1 Assessing Process Capability

Assessing the process capability of the organization starts the deployment process. The overall success of the deployment process depends upon understanding the capabilities of the organization to meet the performance requirements. Process capability measures the ability of the manufacturing to meet the performance requirements. The information provided by the assessment supports the process of deployment, and subsystem partitioning may changes in order to deliver on the requirements.

As an example, if the delivery flow tolerance for a the new drug delivery system is +/- 5% and the current manufacturing line is capable to produce with a +/- 10% tolerance. The assessment has clearly establishes that the manufacturing capability cannot meet the design requirements, and a search for a 3rd party manufacturer begins. After discussions with the 3rd party manufacturer, additional changes to accommodate the new manufacturer may drive changes to the subsystem architecture and associated changes in the design inputs.

This is overall process for assessing process capability follows the following workflow

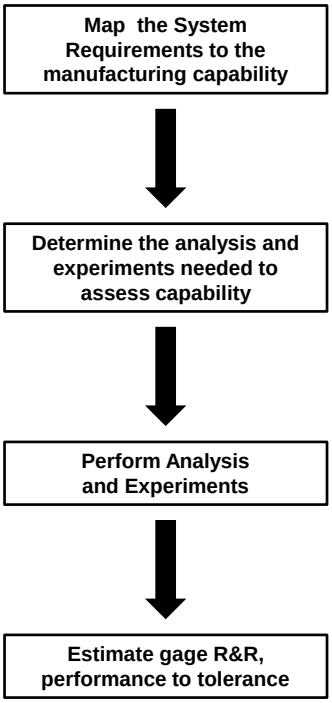


Figure 4-2 Process Capability Assessment

The development of the understanding what can or cannot be done by manufacturing drives the process capability assessment. Process capability information then supports the subsequent development of the design worksheets.

4.1.2 Design Worksheet Development

The Design Worksheet activity creates the detailed mapping from system requirements to the subsystems. During concept development, a preliminary partitioning of the requirements to

the subsystems occurred (section 3.3.3, Subsystem Decomposition), and that partitioning serves as the basis for this more formal and structured mapping.

The process begins with the binning of requirements for subsequent subsystem deployment. The categories for binning are as follows

- **System Performance Requirements Mapped to a Single Subsystem** – these requirements completely map to a single subsystem.
- **Functional Requirements** – these requirements are functional in nature, relating to use cases and other non-measurable requirements
- **Measurable Performance Requirements** – these requirements map to measurable requirements, that is requirements expressed as a value with associated tolerance

System Performance Requirements Mapped to a Single Subsystem

For those system requirements mapped to a single subsystem, process capability assessments against the subsystem determine the validity of the subsystem requirement. Here the entry for the design worksheet entry might be formed as

Table 4-2 Single Subsystem Worksheet

Subsystem	Subsystem Requirement	System Requirement	Deployment	Type	Workflow Diagram	Transfer Function
Delivery	Drug Delivery shall have an accuracy of +/- 5% over the range of 10-1000 ml/hour	Drug Delivery shall have an accuracy of +/- `5% over the range of 10-1000 ml/hour	Single Subsystem	Performance	N/A	N/A

Deployment to a single subsystem does not require workflows and transfer functions, as there are no interactions with other subsystems necessary to fulfill the requirement.

Functional Requirements

For functional requirements, internal interface requirements and use case work flows drive and define the deployment to the subsystems. Workflow (sometimes called sequence) diagrams drive the identification of the interactions and actions of the subsystems.

The following paragraphs illustrate the development a workflow deployment for a complex example. In this example, the initial effort develops an overall sequence diagram, as shown below.

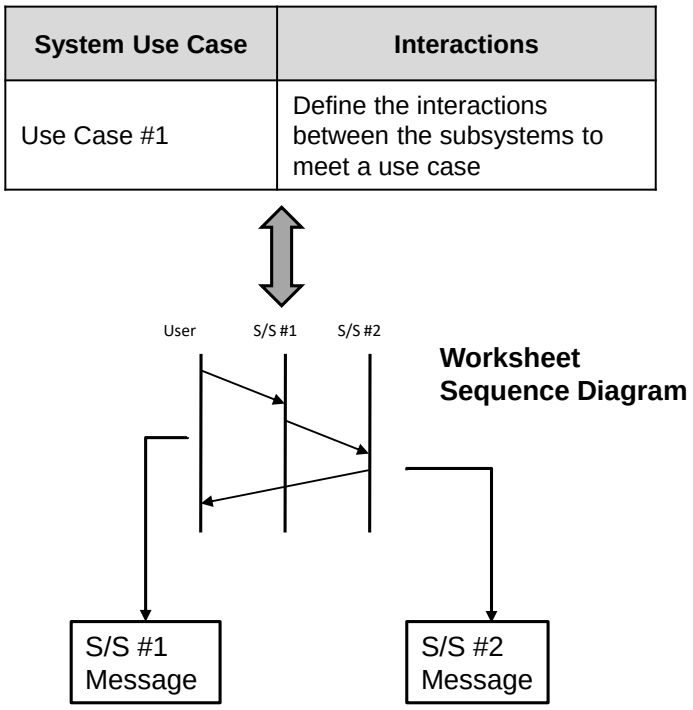


Figure 4-3 Functional Requirement Deployment

The workflow diagram, based upon the internal interface requirements and the preliminary analysis during the concept elaboration format show the interactions between subsystems.

After analysis of the workflow determined that service timings, the time that a subsystem has to respond to (service) a message identified in the workflow diagram, need to be developed. A transfer function approach to the determination of service timing during this step identifies the total service time for a use case. This transfer function details the timing as follows

$$Total\ Service\ Time = \sum message\ timing + \sum service\ timing$$
$$Service\ Time\ Variation\ determined\ early\ start\ versus\ late\ start\ timing$$

The formula above uses the concept of early start versus late start timing from the concepts associated PERT/CPM project scheduling. To determine early and later start timing, the workflow should be constructed as a network, and the critical path can be determined through variations of the timing for the nodes (both message timing and service timing are considered nodes). Variation in the finish time can be found as (Chetan Prakash)

$$Sevice\ Time\ Min\ = \max\{EFT(= EST + maximum\ duration)\ for\ all\ previous\ steps\}$$
$$EFT = Earliest\ Finish\ time$$
$$EST\ = (Earliest\ Predecessor\ Time\ for\ all\ dependent\ events)$$

The following figure shows how the network analysis for a workflow can be structured

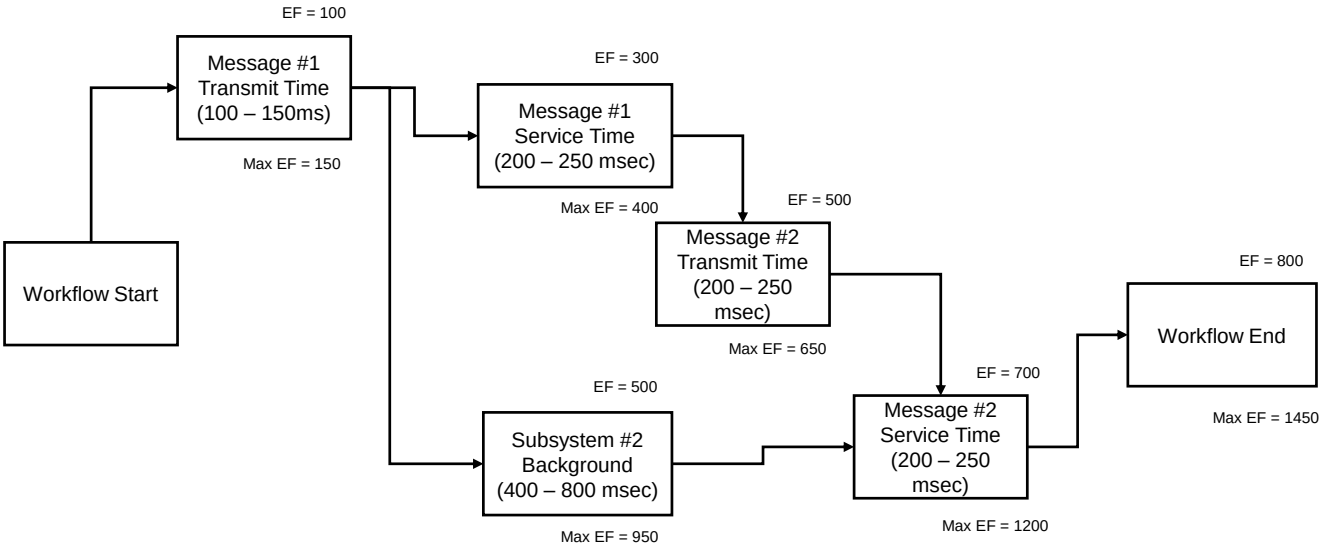


Figure 4-4 Workflow Timing Transfer Function

Many project scheduling programs can perform the calculations shown above, and the use of this tool provides simple way to develop the timing parameters for a workflow

Minimum event timing should also be considered, but this analysis is more heuristic. From the analysis of the network by varying the minimum duration of an event, event timings that occur well ahead of other events can be identified. In this situation, limiting or delaying the event can be considered.

In this example, a final deployment worksheet for this workflow results in a set of complex functional requirements. These requirements rely on the basic workflow diagram as well as a transfer function based upon the PERT/CPM network diagram The following table illustrates the worksheet entries that result from this analysis of a single functional requirement

Table 4-3 Complex Functional Worksheet

Subsystem	Subsystem Requirement	System Requirement	Deploy-ment	Type	Workflow Diagram	Transfer Function
Subsystem #2	Transmit message #1 in 100 to 150 msec	Use Case N	Perfor-mance	Perfor-mance	Worksheet #2	Transfer Function #3
Subsystem #1	Accept Message #1	Use Case N	Functional	Functional	Worksheet #2	N/A
Subsystem #1	Process Message #1 in100 to 150 msec	Use Case N	Perfor-mance	Perfor-mance	Worksheet #2	Transfer Function #3
Subsystem #1	Transmit Message #2	Use Case N	Functional	Functional	Worksheet #2	N/A
Subsystem #1	Transmit Message #2 in 100 to 150 msec	Use Case N	Functional	Perfor-mance	Worksheet #2	N/A
Subsystem #1	Accept Message #2	Use Case N	Functional	Functional	Worksheet #2	N/A

Subsystem	Subsystem Requirement	System Requirement	Deploy-ment	Type	Workflow Diagram	Transfer Function
Subsystem #2	Process Message #2 in 200 to 250 msec	Use Case N	Perfor-mance	Perfor-mance	Worksheet #2	Transfer Function #3

Measurable Performance Requirements

For performance requirement spanning a number of subsystems, worksheet activities involve the development of a formal transfer function. The following figure illustrates the deploym-ment

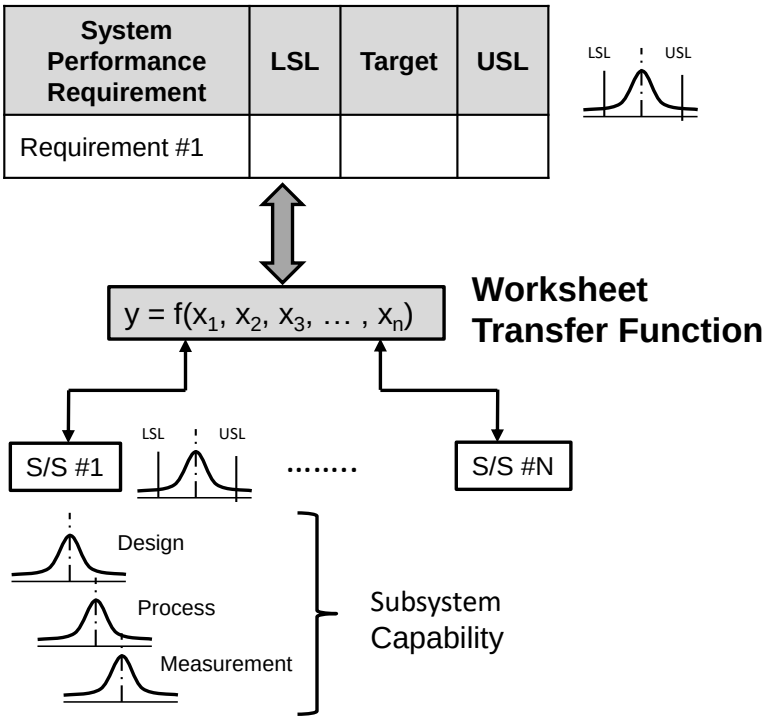


Figure 4-5 System Performance Deployment

In this situation, the performance requirements is broken down by the formula

$$y = f(x_1, x_2, x_3 \ldots x_n)$$
$$where\ y = system\ requirement$$
$$x_n\ is\ the\ deployed\ parameter\ or\ requirement\ for\ subsystem\ n$$

The function $f()$ shows the relationship of the deployed subsystem requirements to the final subsystem requirements.

Performance Transfer functions do not need to be complex. For example, the following transfer function describes the distribution of the system requirement for total weight

$$Total\ weight\ (4kg) = weight_1 + weight_2 + weight_3$$

Development of a transfer function needs to understand the capabilities of the subsystem, and preliminary tolerance/range allocation should be performed during the development of the development of the transfer function. In this example the design worksheet becomes

Table 4-4 Complex Performance Worksheet

Subsystem	Subsystem Requirement	System Requirement	Deployment	Type	Workflow Diagram	Transfer Function
Subsystem #1	Subsystem Weight will not exceed x kg	Total system weight will not exceed 4kg	Transfer Function	Performance	N/A	Transfer Function #1
Subsystem #2	Subsystem Weight will not exceed y kg	Total system weight will not exceed 4kg	Transfer Function	Performance	N/A	Transfer Function #1
Subsystem #1	Subsystem Weight will not exceed z kg	Total system weight will not exceed 4kg	Transfer Function	Performance	N/A	Transfer Function #1

After these worksheet related activities, the subsystem requirements have been established and need to be evaluated. The overall worksheet and the complexity of the resulting subsystem requirements should be critically evaluated based upon the criteria in section 3.3.2.1, Architecture Evaluation Criteria. At the completion of the worksheet development, the system requirements and architecture have been linked in a set of worksheet entries.

4.1.3 Subsystem Requirements Deployment

Subsystem Requirements deployment takes the outputs of the worksheet development and makes a more detailed allocation of these entries across the subsystems. The deployment allocates requirement values and tolerances across the mapped subsystems. During development of the design worksheet, each of the functional or performance requirements was considered independently. This may lead to a situation where worksheet activities for different system requirements generated in conflicting subsystem requirements. Subsystem requirements deployment seeks to rationalize the requirements developed in conjunction with the design worksheets and develop a single, consistent set of subsystem requirements.

As an example, at the completion of the workflow analysis, different timing may be assigned to a single message based upon the service timing of different workflows. The following table illustrates the condition

Table 4-5 Subsystem Requirements Deployment - Timing Example

Subsystem Functional Requirement – Message #1 transmit Time	Timing
From Workflow Diagram #1	100 -150 msec transmit time
From Workflow Diagram #2	200 – 250 msec transmit time
Consolidated S/S Requirements	100-150 msec

The deployment process needs to look carefully at the design worksheet. In many cases, other linked requirements may have an effect, and the change to the subsystem requirement must be carefully considered. In this case, after the review of the workflows, the shorter timing has been adopted, but in another case the longer timing might be a better choice and other requirements adjusted instead. The workflow and network diagrams are critical tools in the deployment process.

For a performance requirement, the process may consider the lower and upper spec limits in addition to the target values. In following example of weight, the consideration of the upper and lower specification limits resulted in tightening the overall range of variation allowed and to accommodate the requirements of the different worksheets.

Table 4-6 Subsystem Requirements Deployment – Performance Example

Subsystem Performance Requirement – System Weight	LSL	Target	USL
From design worksheet #1	3.1	3.3	4.0
From design worksheet #2	2.8	3.1	3.5
Consistent S/S Requirements	3.1	3.3	3.5

4.1.4 The N^2 Chart

Following the completion of the initial subsystem requirements deployment, the final step confirms both the subsystem partitioning and the requirements allocation using an N2 chart . Robert Lano of TRW (Lano, 1977) conceived the N2 chart in 1977, and this concept has been modified and implemented in many different forms, but the key principle is to visually represent the interactions between subsystems.

The N2 chart implementation for subsystems provides a clear presentation of the interaction between identified subsystems. The following figure shows how the N2 chart presents the subsystem interactions

	Subsystem 1	Subsystem 2	Subsystem 3	Subsystem 4
Subsystem 1	No Entry	Interface S/S 1 – 2	Interface S/S 1– 3	Interface S/S 1 – 4
Subsystem 2		No Entry	Interface S/S 2-3	Interface S/S 2-4
Subsystem 3			No Entry	Interface S/S 3-4
Subsystem 4				No Entry

Figure 4-6 Subsystem N^2 Chart

The upper cells identify the worksheet documentation and entries that describe the interactions between two subsystems. In this instantiation, the interface is a reference to the worksheets and transfer functions. The maintenance of the N^2 chart throughout the development process will be critical to subsequent system verification and regression testing.

The N^2 chart represents another sort of the design worksheet. By sorting by the worksheets Diagram and the deployment tables, the information on the coupling of the subsystems can be easily developed.

4.1.5 Requirements Deployment Outputs

The Design Worksheet and the subsystem requirements tables provide information for use throughout the development lifecycle. Specifically, this information provides

- The mapping of how the subsystem requirements fulfill the system requirements
- The interrelationship between the subsystem for each system requirement (through the workflow diagrams, transfer functions and the N^2 chart)
- The rationale for allocations of the requirements
- Completeness for the partitioning of the subsystem requirements

While other approaches can drive similar capability, the design worksheet consolidates information on deployment into a single location, providing a structure for search and analysis capability. Subsystem requirements are just a filter on the design worksheet for the subsystem. Traceability is easily derived from the design worksheet.

It should be noted that the size of a design worksheet often requires the use of a special tool. Sometimes, the use of a simple relational database such as Microsoft Access will be sufficient, and sometimes the scope will require the integration into an advanced lifecycle management tool.

4.2 System Realization

System Realization begins following the deployment of the design inputs to the subsystems. The following table describes the inputs and outputs of the realization process

Table 4-7 System Realization

Inputs	Key Activities	Outputs
System Transfer Functions The details of the interrelationships between requirements, the transfer functions. **Workflow Diagrams** The details of the workflow and subsystem interactions. **Design Worksheet** The details of how the system design inputs and subsystem requirements interrelate **N^2 Chart** The details of the subsystem interactions S**ubsystem Requirements** The subsystem requirements **System Use Cases** Workflows, both primary and secondary **System Requirements** The Design Inputs	**Subsystem Development** The development of the design, manufacturing and service procedures **Subsystem Integration** The planning and execution of the integration between subsystems **Subsystem Integration** The planning and execution of the integration between subsystems **Manufacturing and Service Design** The processes used to manufacture and service the product **Process Characterization** The characterization of the manufacturing and service procedures **System Verification** The verification that the design inputs meet the design outputs. **Regression Development** Response to the issues uncovered in verification	**System Design** The design (including software) for the system **Manufacturing Design** Manufacturing Procedures and test methods. **Service Design** Service Procedures and test methods. **Verification Results** The objective evidence that the system design meets the Design Inputs **Process Characterization Results** The objective evidence that the process can produce the finished device

The following summarizes the activities involved in system realization

- **Subsystem Development** – this involves the development of the subsystem and the associated test methods.
- **Subsystem Integratio**n – the focus here is on the piecewise integration of the system, relying on the worksheet and transfer functions to dictate the staging and integration steps
- **Manufacturing and Service Design** – the interaction with the Development through test methods and the system design
- **Process Characterization** – this identifies which processes require verification and which require validation (this will help considerably during the process qualification)
- **Verification** - the final confirmation of requirements and use cases. This is closely linked to the subsystem deployment (section 4.1)
- **Regression Development** - the incremental adjustments conducted as a result of verification. Again, the subsystem deployment (section 4.1) is critical to the effective execution of regression and associated development

The following figure shows the relationship of these activities

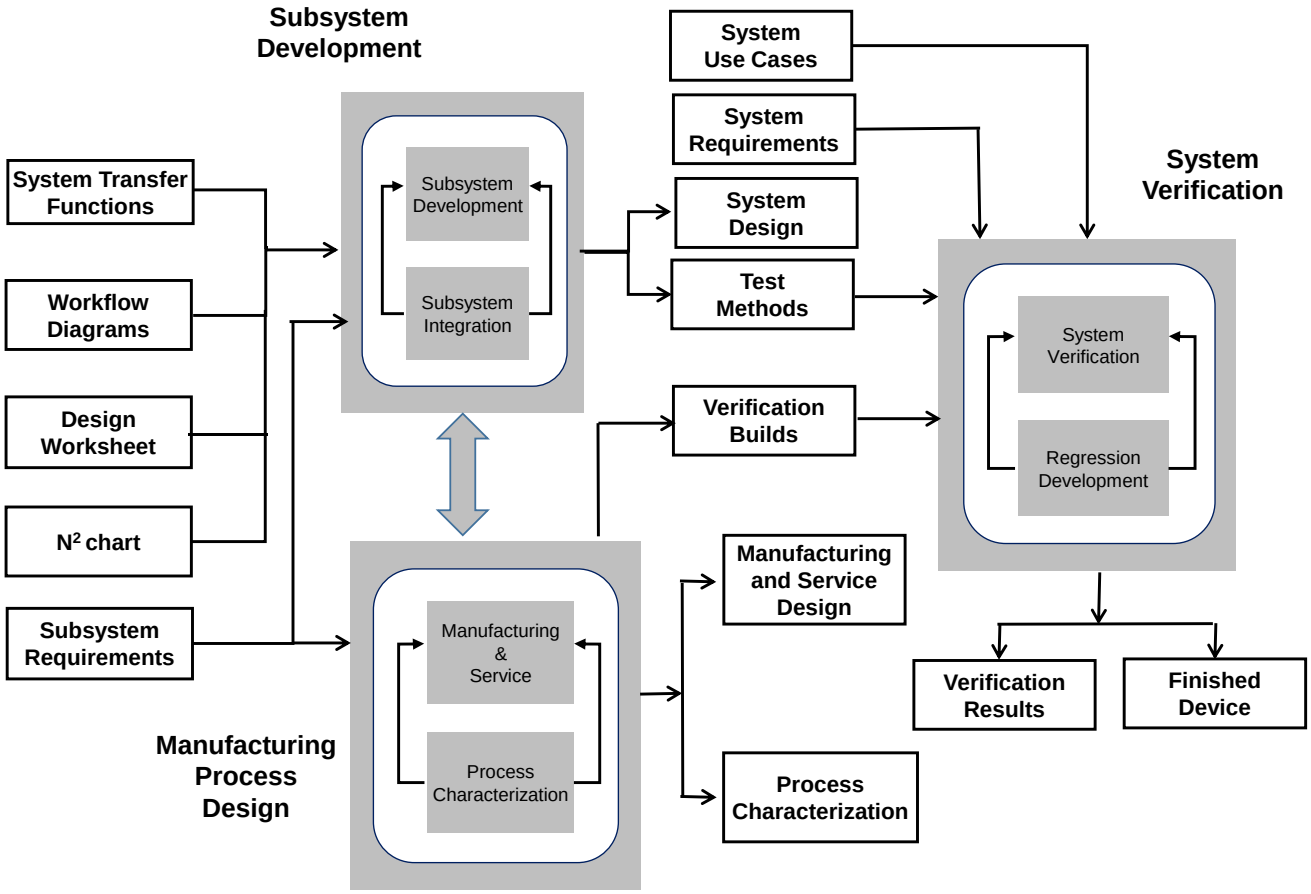


Figure 4-7 System Realization

4.2.1 Subsystem Development

The following section does not attempt to cover the wide range of full scale subsystem development. The concepts and techniques in this section focus upon areas of development where special consideration needs to be taken to ensure no development misses.

Development of each subsystem proceeds based upon the components and technology for that subsystem, but during the development the following needs must be integrated into the activities.

- **Subsystem Verification Design** – analyzing the structure of the requirements to minimize the verification demands of the subsystem.
- **Test Methods** – the empirical testing of the subsystem to confirm that the subsystem meets it performance requirements
- **Subsystem Integration** – the periodic confirmation that the subsystem will integrate with the overall system. Failure to periodically integrate can result in the "boat in the basement" (a boat built in a basement is difficult to get out of the basement without disassembly) situation.
- **User Interface Development** – the development of the user interface is a critical element of many subsystems. A structure process for UI interface development will ensure successful verification an validation at the system level

While many of the above activities could rightfully be performed at the system level, at the subsystem level the activity significantly lowers the creation of "undiscovered rework" that can have significant schedule impact.

4.2.1.1 Subsystem Verification Design

Verification represents a significant effort associated with the development of any medical device, and consideration of testing requirements early in the process can significantly lower the overall effort. Following the deployment of the requirements, consideration of the scope of testing can lead to minimizing the effort and driving more complete testing. Care during initial planning will prevent the development of costly test methods with minimal effectiveness. A tool such as the factorial design associated with design of experiments, when applied early in the subsystem development, can effectively determine the appropriate scope for the subsystem verification.

Leveraging factorial test design initially considers the requirements as factors of the overall subsystem design. During the review of the requirements, review of the deployment of the subsystem requirements can significantly reduce the size of the factorial designs. For example, a full factorial design of 3 requirements with 2 levels (referred to as a 2^3 design (Penn State University College of Health and Human Development, 2015) will require 8 distinct tests with the 3 factors and either the high or low level. If one of the factors does not interact with the other factors this scope could be reduced to a 2^2 design of just 4 tests.

The following steps should be followed to establish the scope of the subsystem verification for performance requirements:

1. Identify which of the requirements represents an input to the subsystem, and which requirement represents an output from the subsystem
2. Use the deployment of the requirements and establish groupings of requirements identified. Grouping should focus on the subsystem output requirements.
3. Identify the requirements for test methods necessary for testing the subsystem
4. Establish the factorial design for each of the groupings

For functional testing, a level of interaction tests should be developed based upon the deployment interactions. This process organizes the functional requirements into groups with dependency and uses this as the basis for the test development.

As an example, assume that we have a subsystem that consists of a circuit board with the following requirements

- The digital signal output shall accurately digitize the input signals over the range of 0-200 millivolts
- The Power supplied to the board shall be 12 volts +/- 0.5 volts
- The board shall weigh 0.6 Kg +/- 0.1 Kg

Clearly, the weight and power input would not be related, but the power and output signal would be related. Signal output has been identified as subsystem performance requirement that will require a test method to be developed. Using the test method developed to evaluate the signal output should have the following range of tests.

Table 4-8 Test Run Example

Test Run	Power	Input signal
1	11.5	0
2	11.5	200
3	12.5	0
4	12.5	200

4.2.1.2 Test Method Development

Throughout the years many have been confused about the needs and requirements for test method validation. Some note that the FDA in the *Code of Federal Regulations (CFR) Title 21 Part 820: Quality System Regulation (QSR)* (U.S Food and Drug Administration) never explicitly mentions method validation, however, it is clear from the first paragraph of Title 21 Part 820 that

> *The requirements in this part govern the **methods** used in, and the facilities and controls used for, the design, manufacture, packaging, labeling, storage, installation, and servicing of all finished devices intended for human use.*

Forgetting about any regulatory ambiguities, it is a good engineering practice to confirm that tests developed to confirm the design performance in development and manufacturing are adequate. In addition, test method definition and development can identify issues with the requirements and the system architecture. In order to prevent rework later in the subsystem development process, test methods should be addressed early in the subsystem development.

A test is commonly defined as “a procedure intended to establish the quality, performance, or reliability of something, especially before it is taken into widespread use”. Key attributes of a test requiring test method qualification include

- Testing that involves using instrumentation or other special test equipment to collect data for later analysis. (Department of Defense, 1994)
- Testing that produces data at discrete points for each specified requirement under controlled conditions and is the most resource-intensive verification technique. (NASA, 2007)

Demonstration is not testing, and test method qualification is not required. Demonstration involves

- The observable operation of the system to confirm a requirement, without requiring instrumentation, special test equipment or subsequent analysis. (Department of Defense, 1994)

This distinction needs to be clearly understood and should drive the subsystem testing accordingly. The significant effort associated with testing should drive the minimization of test methods.

The workflow for the development of a test is as shown in the following diagram (Barwick)

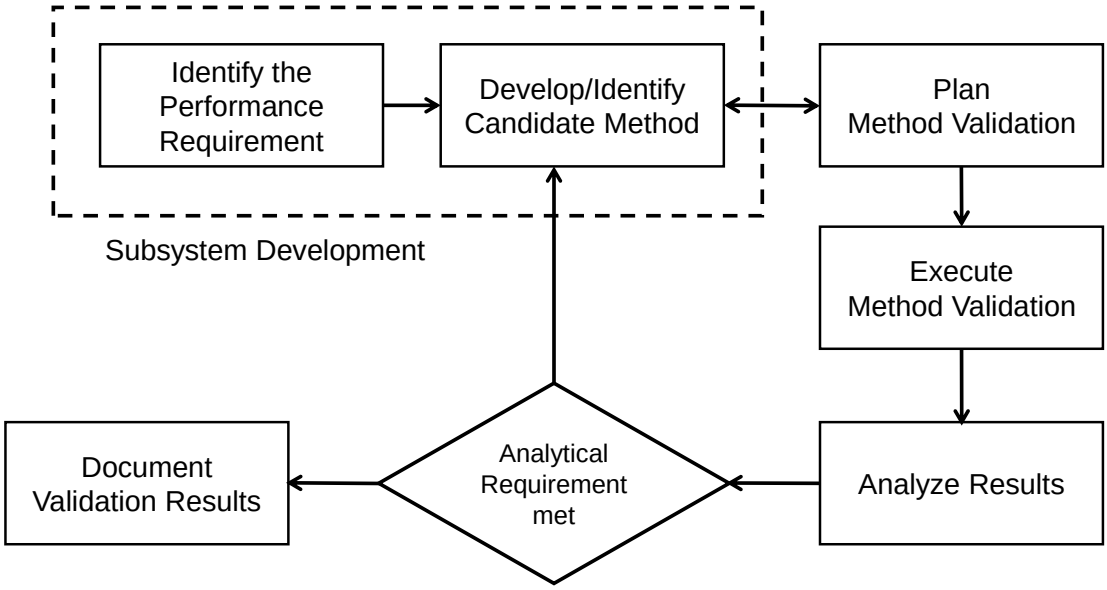


Figure 4-8 Test Method Development

The workflow identified here is iterative throughout the subsystem development. Consideration of the method validation usually takes place in parallel with the identification of the test method, and usually there will be iterations between the plan and the method, prior to the execution.

A variation of the P-diagram ((Phadke, n.d.) targeted at test method development represents a critical tool in the development of the candidate test method. The following diagram shows the basic structure of this variation

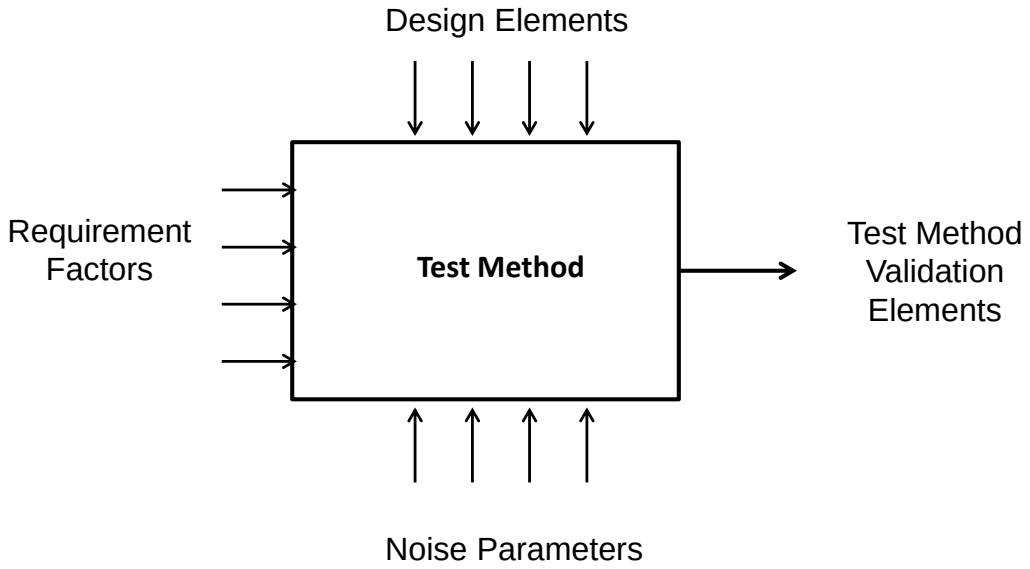


Figure 4-9 Test Method P-Diagram

In this case the focus is not on the design of the system, but the test method. The elements of this variation are as follows

- **Requirement Factors** - The range of the factors associated with the requirement. For performance requirements such as delivery accuracy, this may include the full range of the operation for the device. The initial analysis (section 4.2.1.1Subsystem Verification Design) provides the initial inputs to this process
- **Control parameters** - The elements of the design that are the key contributors to the requirement. These are the design specifications.
- **Noise** – The elements of design and development that cannot be controlled by the design process.

The method development process begins with the identification of the factors. Often a specification is of the form

Pressure measurement accuracy shall be +/- 5% over the range of 0-100 kPa

In this case, the key factor of the test method is a single parameter, the pressure, with a range of 0-100 kPa. If a requirement contains a statement such as “normal operating conditions”, the statement should be replaced with specific conditions related to the temporal, environmental and operating ranges.

Derivation of the control parameters requires the analysis of the subsystem(s) that deliver on the requirement. Typically a design review can quickly develop this list of control parameters.

Noise for a test method addresses the Repeatability and Reproducibility, commonly referred to as “gauge R&R”. In the context of a test method

- **Repeatability** - the variation in measurements taken by a single person or instrument on the same or replicate item and under the same conditions.
- **Reproducibility** - the variation induced when different operators, instruments, or laboratories measure the same or replicate specimen.

Discussions of measuring gauge R&R will not be discussed, but without this analysis development of a test method’s precision will be difficult.

Other noise elements for the test also need to be considered. Again, a design review can identify other noise elements inherent in the design and operation.

Following the identification of the elements, controls and noise, the development of the method must consider the following elements

- **Specifications** – this should include the reference values as well as the Lower Specification Limit (LSL) and Upper Specification Limit (USL). Additional specifications may include the AQL (acceptable quality level) of the test when used for manufacturing qualification.

- **Accuracy** – what is the overall accuracy of the test, that is does the measured value reflect the reference value is required, and can the candidate test method meet that requirement. Performance requirements generally specify a tolerance that may serve as the basis of this element
- **Precision** – how precise is the test. Usually Repeatability and Reproducibility, sometimes in the form of a Gauge R&R determines the precision
- **Range** – the range or continuum over which the test method would be considered accurate. This should be at least the range of the performance requirement.

The relationships between these parameters are shown in the following figure.

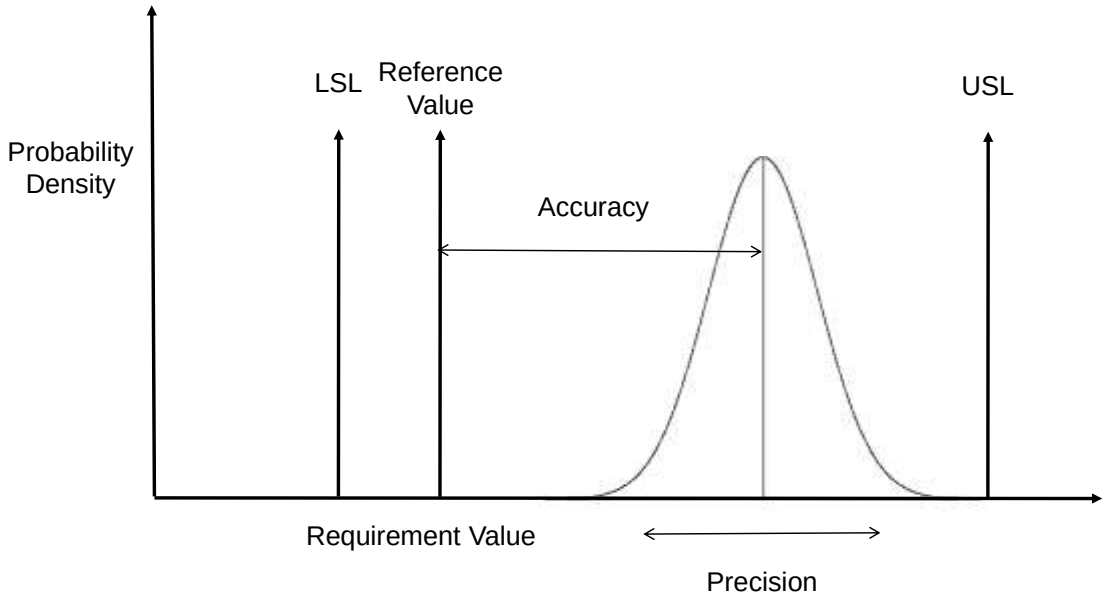


Figure 4-10 Relationship of Specifications, Precision and Accuracy

Both accuracy and precision determine the validity of the testing both the accuracy and the precision of the test need to be considered. The following figure shows the relationship between accuracy, precision and the validity of the test

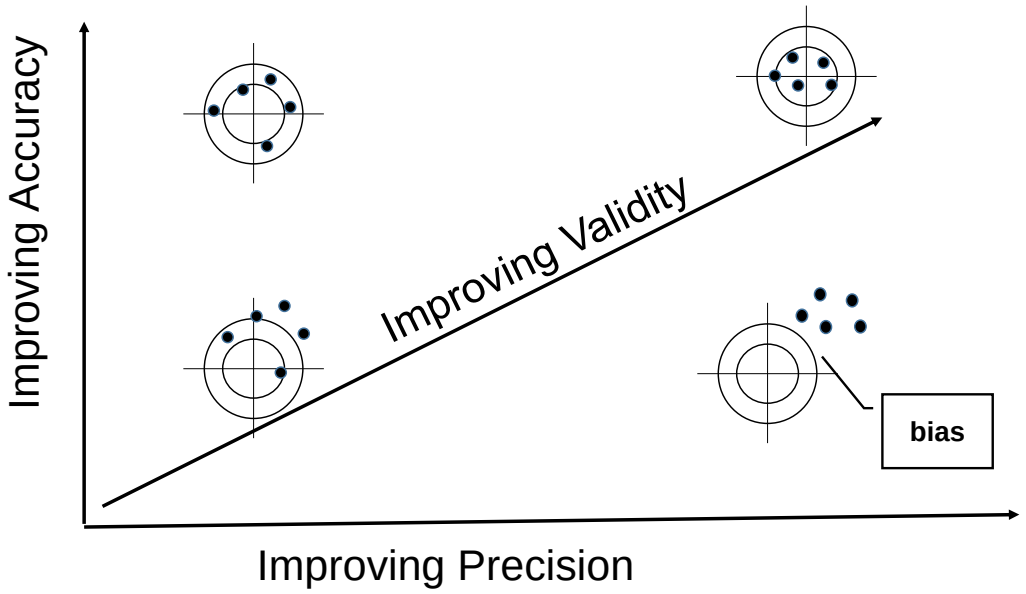


Figure 4-11 Accuracy in Testing

A test can be very precise, but elements such as bias can impact the accuracy. Conversely, a test can be accurate, but the precision can impact the quality of the tests.

When considering the accuracy and the precision requirements for the test, manufacturing test need to consider the appropriate AQL (Acceptable Quality Level) (ASQ, 2013). The details of AQL will not be discussed, but this should be a key element of developing the accuracy and precision needs for the test. AQL should be strictly linked to the risk associated with the requirement. In cases where the requirement directly links to risk mitigation the AQL should be higher. AQL for tests and requirements not associated with risk mitigations can be relaxed and simpler tests can often be developed.

Once the elements of the test method P-diagram have been identified, the test method validation plan can be developed. Key techniques such as Design of Experiments (DOE) should be employed to establish the full range of the testing required to ensure that the method is valid.

Development of test methods takes considerable effort during system realization. Prior to full scale subsystem development, the consideration of the effort demands consideration of the test methods. The key considerations should be

- **Minimize the number of tests** – during concept deployment (section 3.3.6) the consideration of the nature of the verification (Inspection, Analysis, Demonstration, Test) needs to be revisited and evaluated
- **Ensure that the test requirements are achievable** – understand that specification limits, AQL and range will impact the complexity of the testing and that over specification of these requirements can result in a significant increase in the effort.
- **Plan for test method development and validation** – this is a significant effort and needs to be accounted for as a critical part of development

- **Initiate test method efforts concurrent with the development** – don't wait till the end of the development cycle to address test methods.

4.2.1.3 Wireframe UI Development

Another area of significant rework during the development of the subsystems is the development of user interface (UI) elements. In developing the UI, the lack of a structured process often leads to significant iteration and rework. Use of a Wireframe process can significantly reduce or eliminate the rework associated with UI development.

The key workflow for structured wireframe development

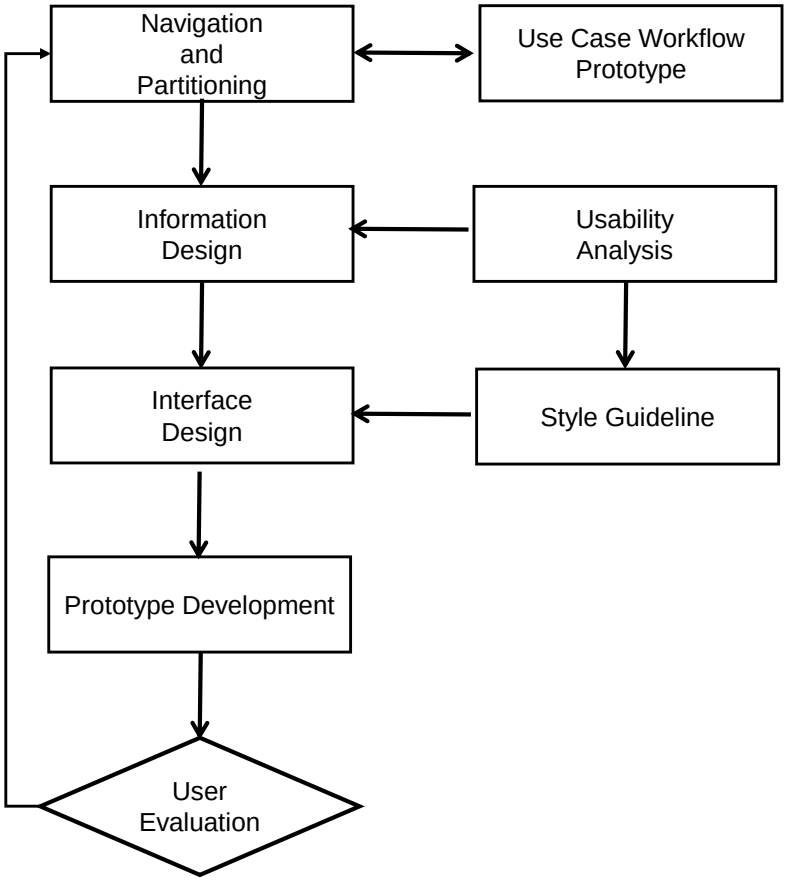


Figure 4-12 Wireframe UI Workflow

This workflow clearly separates development activities to ensure consistency and completeness. The following SIPOC describes the process in more detail

Table 4-9 Wireframe SIPOC

Step	Inputs	Processing	Outputs
Navigation and Partitioning	• System Use Cases • User Needs • General Tool Capabilities • Use Case Workflow Prototype	• Review System Use Cases and User Needs • Develop the navigation and screen partitioning	• Documented Navigation Document
Information Design	• Documented Navigation Document • Usability Analysis	• For each screen develop the information model	• Information Model
Interface Design	• Style Guidelines • Implementation Guidelines • Information Model • Navigation • System Use Cases – Alternate Flows	• Develop the look, feel and interaction portions of the wire frame • Develop exception and alternate workflow handling	• Wireframe Document
Prototype Development	Wireframe Document	• Prototype	• Wireframe Model

Navigation and Partitioning utilizes the use cases and a preliminary workflow prototype to develop the workflow. At this stage, the expectation is that the team works with the users to develop the workflow based upon the clinical workflow. The prototype should not explicitly detail interactions at the screen or field level, but focus on the flow from screen to screen. The workflows developed as part of the risk analysis (section **Error! Reference source not found.**, **Error! Reference source not found.**) from can be a good input to the development of the workflow. At the end of the navigation and partitioning step, the screens have been defined and the flow between the screens has been defined, but the content of the screens are not defined.

Information design details the information elements of the user interface. The following is an example of information design

Table 4-10 Information Design

Data Element	Screen/Interface	Type	Range/Constraints	Proposed Entry Mechanism
Dose	Set Dose Screen	• Floating Point	• 0-200 PPM • 0.1 PPM Range	• Knob • Up/down arrows
Patient Name	Start Screen	• Text	• Up to 45 characters	• Text typing (touch keyboard)

Usability analysis is based upon the following:

- The nature of the use environment
- The characteristics of the user
- The nature of the overall use

This activity identifies the types of user interface widgets that may be applicable to a particular informational element. As an example, the use of certain standard user entry elements may not be appropriate if the use environment is a surgical or critical care environment. Usability analysis provides the basis for entry mechanism selection.

The next step structures the placement of interface elements for each of the screens identified in the partitioning. Style guidelines help guide the overall placement of information and flow control elements. Development of the screens needs to consider the use and user preferences in placement, placing interface elements so that the most used elements are consistently placed on the screens (usually at the top). Upon completion of this step, each screen has a detailed design document that documents

- **Element Layout** – The placement on the screen
- **Individual Element Validation** – The validation rules for each individual element. This should include the on/off states for each button or navigation element
- **Screen Level Validation** – The validation rules for the screen. This should identify the rules to be applied to element values following the completion (submission) of the screen.

Often teams skip the formal documentation during UI development, but this can negatively impact the testing and support of the UI during iterations. The step to formally document will yield benefits during the iterations and later sustaining support activities.

Only after the completion of the complete screens should the development of the prototype begin. The documentation developed above will allow the development of the user interface prototype. Typically the validation elements of the UI can be omitted from the prototype, but certainly layout and workflow are completed. With the prototype the user interface is presented for evaluation and adjustments are identified.

4.2.1.4 Subsystem Integration

The key to effective subsystem integration is to integrate functionality early in the development of the subsystem. To implement effectively, other subsystems need to be at a similar state of readiness. Coordinating schedules between subsystems requires reviewing the N^2 chart developed during deployment to understand opportunities for early integration. As shown in the following figure, review of the N^2

Subsystem 3
Integration Candidates

	Subsystem 1	Subsystem 2	Subsystem 3	Subsystem 4
Subsystem 1	No Entry	Interface S/S 1 – 2	Interface S/S 1– 3	Interface S/S 1 – 4
Subsystem 2		No Entry	Interface S/S 2-3	Interface S/S 2-4
Subsystem 3			No Entry	Interface S/S 3-4
Subsystem 4				No Entry

Figure 4-13 Subsystem Integration with N^2 chart

Following the analysis of potential early integration opportunities, a comprehensive subsystem development schedule can be developed. This schedule should stress early integration of subsystems. It has been noted ((Grant E Head, 1994) that during system development most of the design flaws occur during the integration of system elements. To this end, the early identification of these issues has a great deal of impact upon the overall performance to schedule.

4.2.2 System Verification

System Verification follows the verification and integration of the subsystems. At the system level, the initial activities focus on understanding and testing of the subsystem interactions as described in the performance and functional requirements.

Requirements Deployment (section 4.1.2, Design Worksheet Development) drives the verification plan. The requirements were categorized as follows

- **System Performance Requirements Mapped to a Single Subsystem** – these requirements completely map to a single subsystem.

- **Functional Requirements** – these requirements are functional in nature, relating to use cases and other non-measureable requirements
- **Measureable Performance Requirements** – these requirements map to measureable requirements, that is requirements expressed as a value with associated tolerance

Those system performance requirements mapped to a single subsystem have already been verified by the subsystem verification activities and no further system verification is needed. Functional and Performance requirements not mapped to a single subsystem need to be verified at the system level.

The simple verification of the system level requirements is not sufficient to verify the system. Critical to proper operation of the system is the testing of the interactions between the elements of the systems. Tools such as pairwise testing are essential tools in the testing of the interactions between subsystems and use cases.

4.2.2.1 Functional Requirements Verification

Functional verification starts with the functional requirements, and consists of a set of tests structured to confirm these functional specifications. This testing will rely on the testing of the both the main success scenario as well as the failure modes. As noted in section **Error! Reference source not found.**, **Error! Reference source not found.** and shown in the figure (repeated here) the full testing of the use case will involve the execution of the main success scenario as well as the failure mode extension. This structure allows the full development of the testing for the functional requirements mapped to the use case or workflow.

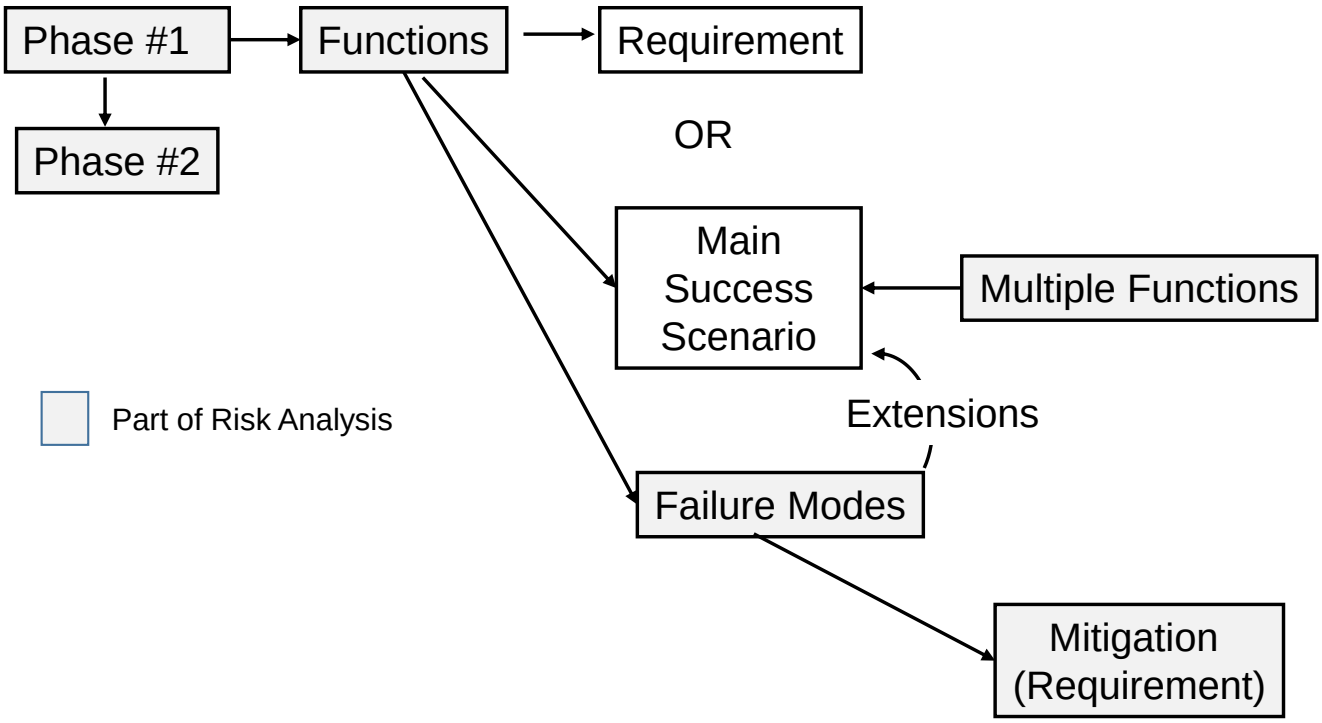


Figure 4-14 Use Case Linkages

During the Design Worksheet Deployment, the use cases were elaborated as associated sequence diagrams (see section 4.1.2, Design Worksheet Development.) Within the sequence diagram, additional performance requirements were linked to the sequence, and the performance testing can either structured as a separate verification test, or a combined functional/performance test developed. The testing will link to each of the associated requirements through the Design Worksheet.

4.2.2.2 Performance Verification

Performance verification follows the same pattern as subsystem performance verification testing. Test methods are developed and confirmed (see section 4.2.1.2, Test Method Development)

4.2.2.3 System Interaction Testing

Interaction testing at the system verification level seeks to confirm that the overall system operates well and that the interaction of the subsystems, uses cases and sequences does not create issues. As an example, memory leaks in software can create significant issues, but the execution of a single functional test may not identify this type of issue. As another example, a parameter set by the user in a setup workflow may manifest itself during the alarm handling operations. Pairwise testing seeks to identify these issues.

In 1994, in an article on cleanroom software engineering (Head, 1994), it was noted that almost all defects in software systems were detected during integration testing. These defects can be traced to misunderstanding and miscommunication between development groups.

The realization that defects related to system interactions are difficult to find and can be expensive to correct in a released product have led to the use of combinatorial testing, with the most common combinatorial testing being pairwise testing. In pairwise testing, a set of pairwise interactions between the software is constructed using standard design of experiment techniques.

As shown in the following figure ((D. Richard Kuhn, 2004), pairwise testing associated with medical devices can detect well over 90% of the defects within the system.

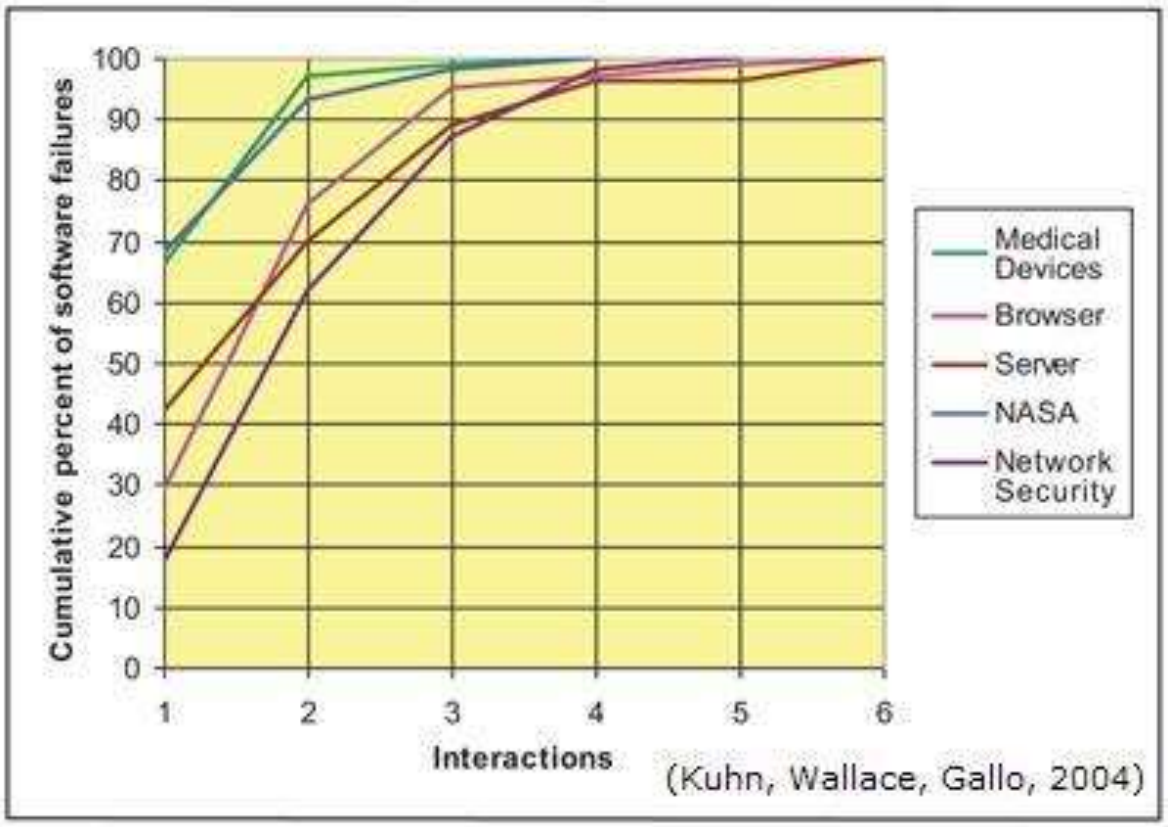


Figure 4-15 Combinatorial Testing Effectiveness

The process of begins by leveraging the N^2 chart developed earlier (see section 4.1.4The N^2 Chart). The chart is repeated here

	Subsystem 1	Subsystem 2	Subsystem 3	Subsystem 4
Subsystem 1	No Entry	Interface S/S 1 – 2	Interface S/S 1– 3	Interface S/S 1 – 4
Subsystem 2		No Entry	Interface S/S 2-3	Interface S/S 2-4
Subsystem 3			No Entry	Interface S/S 3-4
Subsystem 4				No Entry

Figure 4-16 N^2 Chart

Using this chart, the interactions that will require the development of pairwise testing can easily be developed. From the use case development we can develop the matrix of interactions between the main success scenarios and extensions for the identified use cases. The structure of this analysis is shown in the following figure

	Subsystem # 2 Main Path Scenario	**Subsystem # 2 Extension Scenario # 1**	**.....**	**Subsystem # 2 Extension Scenario # N**
Subsystem # 1 Main Path Scenario	**X**	**X**		
Subsystem # 1 Extension Scenario # 1			**X**	
.....				
Subsystem # 1 Extension Scenario # N				

Figure 4-17 Pairwise Testing Identification

At this point, the applicable tests for each pairwise test have been developed. Execution of these tests should represent complete interaction testing.

4.2.2.4 System Regression Testing

During Verification testing, defects will be identified and corrected. The correction of a defect

can be related directly to the failure to meet a requirement of a failure arising from the interaction of the system functions. Based upon the structured deployment of requirements and the pairwise testing analysis the following steps can be applied to find the minimum regression testing set of tests.

1. Use the Design Worksheet to identify the use cases and transfer functions associated with the failed requirement. Tests linked to the use cases and transfer functions represent the first level of testing
2. Use the N^2 chart to identify the interactions and use the associated pairwise testing

This mechanism will ensure the completeness of any testing. If multiple defects are fixed the minimum set of testing can be expanded to the set of testing for each defect, ignoring tests that are repeated in each analysis.

4.3 Product Validation

Product Validation confirms that the product meets the user needs and intended uses, that is, is the finished device the right product. Product Validation involves the testing of the device in a simulated use environment using a production equivalent device to confirm that the device meets the user needs and intended uses.

Section 3, Creating the Requirements, clearly established the user needs. Within section **Error! Reference source not found.**, **Error! Reference source not found.**, the flow of the user actions has been detailed, and this flow will serve as the basis of subsequent testing. User need statements established in section 3.1, Concept Selection form the basis of the user needs evaluation.

Validation requires a structured approach with careful planning. A simulated use environment and the results from testing in this environment can be much more variable than verification testing, and without careful attention to the aspects of the testing, the results will not confirm the intended uses and user needs. Any failures at this point in the process will be costly, will result in increased regulatory scrutiny (and possible failure to achieve regulatory approval), and will cast doubt on the entire development process.

The following table describes the inputs and outputs of the Validation Process

Table 4-11 Validation SIPOC

Inputs	Key Activities	Outputs
Use Needs The key user needs and the associated levels of customer/user satisfaction	**Validation Planning** The scenarios and participant requirements and planning associated with the validation	**Final Validation Report** The final report detailing the validation of the system details of the transfer

Inputs	Key Activities	Outputs
Use Cases Workflows, both primary and secondary **Hazard Analysis** The hazard analysis of the product **System** The completed and verified product	of the products **Validation Execution** The execution of the scenarios and compilation of the results associated with system validation **Validation Analysis** Develop transfer functions from results of the Design worksheet development	function. **Final, Validated System** The final system

4.3.1 Validation Planning

Validation planning should begin as soon as requirements are established.

The first step in validation planning is identification and recruitment of the validation participants for the simulated use environment. Typically, the activities associated with establishing a simulated use test and enrolling simulated use participants is a long-lead item, requiring significant time. The recruitment process must identify right composition and number of participants. The following rules should be applied for the recruitment (Kaye, Ron,, 2010)

- At least 15 or more participants should be involved. These participants should be representative of the user population.
- Members of the design team and other employees of the company should not be considered

Additional guidelines are outlined in the FDA's document "Applying Human Factors and Usability Engineering to Optimize Medical Device Design" (U.S. Department of Health and Human Services, Food and Drug Adminstration, 2011). The identification or definition of the "representative user" must be given careful consideration. Too general a definition may impact the results due to the recruitment of participants with inappropriate understanding of the device and its use. Frequent reviews and interaction with the team or firm responsible for the actual recruitment should take place to get the right "representative user".

Following the identification of the "representative user", the identification of the simulated use scenarios begins. Through the risk analysis (section3.2, Risk Analysis) the phases and functions associated with the use of the device have been identified. The phases and the

sequence of functions can be used to structure the top level use scenarios. As an example, a use scenario may be the phase “program the device”. Further analysis of the alternate flows (section **Error! Reference source not found.**, **Error! Reference source not found.**) can identify if a given phase requires more than one scenario.

Upon completion of the scenario identification, the potential use error failure modes and mitigations need to be analyzed. Setup of the scenario should take into consideration the use error taxonomy outlined in section 3.2.4, Risk Mitigation and Controls . The following elements of this taxonomy need to be evaluated

- **Noise and Distraction** - Use errors characterized as slips can often be caused by noise and distractions caused by the environment, and an evaluation should consider if the potential for these types of use errors is adequately reflected in the scenario outline.
- **Training** - Use errors based upon lapses are often related to training and familiarity with the device/environment. To properly assess the ability of the simulated use to identify and evaluate use errors based upon lapses an evaluation of the training should be considered. Too little participant training will result is lapses that may not reflect the actual standards of training, and too much participant training may skew the results, resulting in a fewer lapses than might be encountered in real use.
- **Experience** – the evaluation should consider whether or not the participants represent the overall skill level of the user population. Too much experience may not reflect the impact of lapses, while too little experience may skew the results in a negative manner.

In order to fully validate the satisfaction of user needs, a Likert scale questionnaire for establishing the ability of the system to meet the established user needs is required. These needs, as established in section 3.1.1, VOC Development) will be confirmed by surveying the validation participants at the end of the validation scenario testing.

4.3.2 Validation Execution

The execution of validation should follow the established plans and scenarios as closely as possible. As part of the execution of activities associated with the use scenarios and simulation, the activities of the participants should be filmed. Filming will help considerably in the understanding of issues that may arise during the simulation execution.

Following each participant’s session, a general discussion and survey should be conducted to understand the issues encountered during the session. User comments should be recorded accurately, with no attempt to paraphrase or apply analysis to the comments. The accuracy and completeness of the comments will be critical in developing the validation assessment.

In addition to discussions on issues, each participant should complete the Likert scale questionnaire developed as part of the planning. This will support the analysis of the systems’ ability to meet the defined user needs.

4.3.3 Validation Analysis

Following the execution of validation, the issues encountered need to be evaluated and dispositioning of each issue documented. Validation execution of use cases, by its nature, creates issues, and the analysis and dispositioning of these issues will determine if the complete system meets the user needs and intended uses.

Analysis of an issue starts with an attempt to understand if the issue was caused by the testing environment or the system performance. An issue or use error found during validation that has can be mapped to a hazardous situation analysis cannot be attributed to the simulation testing scenario execution (but the frequency of occurrence may be a result of the test environment).

For issues and errors that cannot be traced back to the hazardous situation analysis, analysis to ascertain if the situation is an artifact of the scenario test environment not identified in earlier analysis. If analysis indicates that the issue was due to the testing environment, an evaluation of other testing scenarios needs to be conducted. This analysis should confirm that other testing scenarios are not susceptible to a similar condition.

For validation issues where the situation cannot be mapped to the hazardous situation analysis, and no testing issue has been identified, a risk analysis must be conducted. This analysis should determine the overall frequency and severity of the issue and the need for mitigation, in accordance with the risk analysis detailed in section 3.2, Risk Analysis. The result of the risk analysis may require system redesign and some repetition of the validation testing.

Frequency determinations conducted as part of the analysis of validation issues should not be based upon the observed frequency. Sample sizes associated with validation do not support use of the “raw” frequency of occurrences divided by opportunities. The analysis of a validation issue with respect to validation needs to answer the question, “Does the observed number of occurrences indicate that the frequency in the risk analysis is incorrect?” In place of the “raw” frequency to answer this question, hypothesis testing should be used.

The null hypothesis for the assessing the issues rates seen in validation are that the frequency is different than the estimated frequency of the hazardous situation analysis. A two population test should be used. One population is the validation occurrences versus opportunities and the other population is the estimated frequency from the risk analysis. For example, if validation experiences one user error in 30 occurrences during validation and the projected frequency in the hazard analysis is 0.001 (1 in 1000) the two population grid is

Table 4-12 Two Population Distributions

	Population Distributions	
	Failures	Successes
Validation	1	29
Risk Analysis	1	999

In this example, the Fisher's exact test statistic value is 0.057432. The result is not significant at $p < .05$. This means that this frequency in the testing cannot be stated as different frequency from the hazard analysis. Note that had 2 errors occurred the statistic would have been significant and the observed frequency could be considered different than the hazard analysis.

This form of hypothesis testing can quickly determine whether or not the validation testing has confirmed the system is operating within its risk profiles. If the system is operating within the risk profile, no further action is required. If there are issues where the validation indicates that the frequency of occurrence (or the severity) is not aligned with the hazard analysis, additional hazard analysis must be undertaken. Again, this analysis should be in accordance with the risk analysis detailed in section 3.2, Risk Analysis.

If the validation confirms the risk profile, the final analysis step confirms that the system meets the user needs. In order to correctly understand the results, each participant should again be asked to rank the best-in-class for each of the user needs. Statistical tests such as the Wilcoxon signed ranked test can be used to compare the data from the early confirmation surveys to the validation results, but more often than not, a simple heuristic analysis will be sufficient. A heuristic test may consider the distribution of the distance from best-in-class for the surveys, comparing the results from concept confirmation to the results obtained during validation.

Following the analysis of the validation results, a complete report should be compiled that details the

5 Program Management and Device Development

5.1 Managing the DHF and the DMR

5.2 Schedule Management

5.3 Design Reviews

6 Summary

As a reader digests through the processes and techniques noted here, a common reaction is "Do I need to do all this work". In an environment where "time is money", the desire to short cut the steps is unavoidable, but as shown in the figure below, the interconnections of the work outputs are such that mistakes and omissions made in the earlier steps can have severe impacts in later stages.

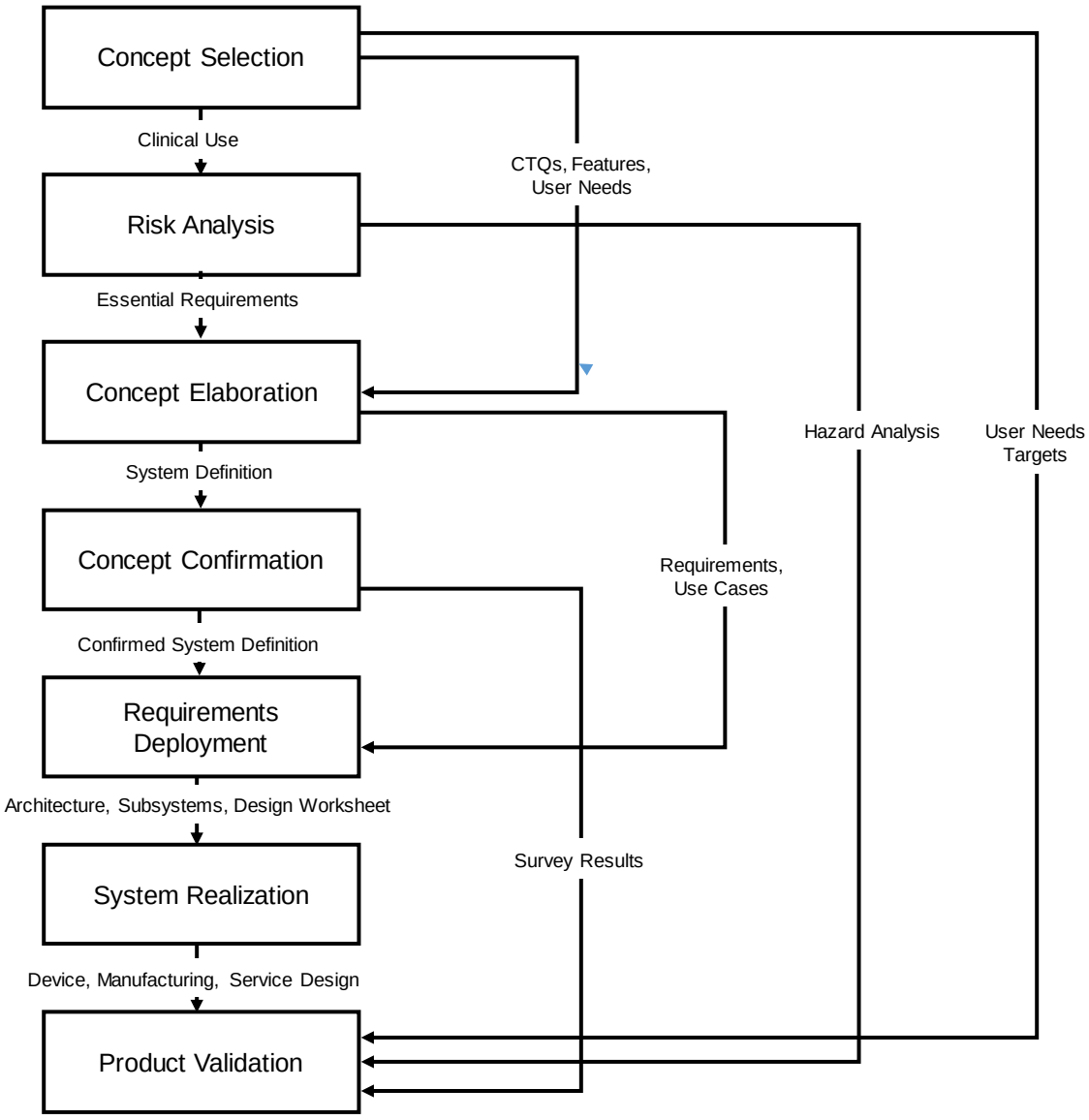


Figure 6-1Step Interactions

A study undertaken by NASA in 2004 (JM Stecklein, 2004) showed that errors in late stages can be more than 50 times as expensive to fix as errors early development stages. In addition, the propagation of undiscovered rework has been shown (Pugh G. P., 1981) to be the single biggest cause of system development delays. With these considerations, the key question is not whether to perform the work, but how to perform the work efficiently and effectively.

In order perform the work efficiently and effectively, the key element is the concept of “good enough”. The “good enough” evaluation is conducted in during next step of the process. If the flow down from the previous step makes the current step difficult, the previous step should be revisited and reworked. In this manner, only a single step will be repeated and the impacts will be mitigated. Additionally, the tables and structures here are generally simple and often do not require additional information. Adding complexity will not necessarily add clarity.

As an example of scaling, the concept of a sequence diagram and timing analysis should consider if the timing associated with the sequence is critical. An initial analysis of the sequence and the architecture should be undertaken and if the heuristic analysis indicates that timing is not an issue, this should be documented and timing need not be performed. It is critical in these situations to document the assumptions associated with skipping the formal analysis, so that if rework is required, the rationale for the original work is clearly detailed. Failure to document these rationales can slow down the rework process and can create doom loops of continual updates.

As another example of scaling, most development for medical devices involves upgrades to a subset of the subsystems, not complete development. If the design worksheets and subsystem flow down has been performed correctly, the impacts to the system can be clearly identified. The linkage to subsystems can be identified through the flow down, and the scope of the impacts limited to the extent of the changes. This will prevent the rework associated with missing critical changes during the later phases of the project.

Finally, nothing supports efficiency and effectivity more than repetition. As this becomes the way the team works, the efficiency will increase. The team will understand the level of detail required based upon experience and the knowledge of the system. Nothing in this process is inherently difficult, but the first time for everything takes more time. A development team should continually avoid the temptation to abandon the process. The simple fact is that the process will learning involves mistakes and rework, and this process is how a team learns. Initial planning for first steps should incorporate sufficient time to account for rework and mistakes.

7 Bibliography

Administration, U. D. (2016, February 3). *Applying Human Factors and Usability Engineering to Medical Devices: Guidance for Industry and Food and Drug Administration Staff.* Retrieved from Food and Drug Adminstration: http://www.fda.gov/downloads/MedicalDevices/.../UCM259760.pdf

ASQ. (2013). ANSI/ASQ Z1.4–2003 (R2013): Sampling Procedures and Tables for Inspection by Attributes.

Barba, R. J. (2004). Failure Mode Effect Analysis Applied to the Use of Infusion Pumps. *Proceedings of the 26th Annual International Conference of the IEEE EMBS.* IEEE.

Barwick, V. (n.d.). *Introduction to Method Validation.* Retrieved from Royal Society of Chemistry: http://www.rsc.org/images/Warwick2_tcm18-93083.pdf

Bettencourt, A. W. (Spring 2008). Giving Customers a Fair Hearing. *MIT Sloan Management Review*, Vol 49 NO 3.

Chapman, R. (2012, July 20). *Assurance Cases For External Infustion Pumps Update*. Retrieved from FDA: https://www.umsec.umn.edu/sites/www.umsec.umn.edu/files/AssuranceCases2012-UMSEC-SSS.pdf

Chetan Prakash. (n.d.). *Department of Mathematics, California State University San Bernadino*. Retrieved from Project Scheduling: PERT/CPM:

http://www.math.csusb.edu/faculty/prakash/611/Project_Management.ppt.pdf
Cockburn, A. (1998). *Basic Use Case Template*. Retrieved from Alistair Cockburn: http://alistair.cockburn.us/Basic+use+case+template
D. Richard Kuhn, D. R. (2004). Software Fault Interactions and Implications for Software Testing. *IEEE Transactions on Software Engineering*. Retrieved from NI: http://csrc.nist.gov/groups/SNS/acts/index.html
Department of Defense. (1994). *DI-IPSC-81433 SOFTWARE REQUIREMENTS SPECIFICATION (SRS).* DOD.
FDA. (2014, June). *UCM404248*. Retrieved from FDA: http://www.fda.gov/downloads/Training/CourseMaterialsforEducators/NationalMedicalDeviceCurriculum/UCM404248.pdf
Fishbone (Ishikawa) Diagram. (n.d.). Retrieved from ASQ Knowledge Center: http://asq.org/learn-about-quality/cause-analysis-tools/overview/fishbone.html
Grant E Head. (1994). Six Sigma Software Using Cleanroom Software Engineering Techniques. *Hewlett Packard Journal*, 40-50.
Head, G. E. (1994). Six-Sigma Software Using Cleanroom Software Engineering Techniques. *Hewlett-Packard Journal*, 40-50.
IEC. (2006). *IEC 60812 Analysis techniques for system reliability – Procedure for failure mode and effects analysis (FMEA.* IEC.
IEC. (2015). *IEC 62366-1:2015 Medical devices -- Part 1: Application of usability engineering to medical devices.* ISO.
ISO. (2012). *BS EN ISO 14971:2012.* ISO.
ISO/IEC. (2014). *ISO/IEC 90003:2014Software engineering — Guidelines for the application of ISO 9001:2008 to computer software.* BS ISO/IEC.
ITU-T. (2008). *X.680 SERIES X: DATA NETWORKS, OPEN SYSTEM OSI networking and system aspects – Abstract Syntax.* ITU-T.
JM Stecklein, J. D. (2004, June 19). *NASA Technical Report Server (NTRS).* Retrieved from Error Cost Escalation Through the Project Life Cycle: http://ntrs.nasa.gov/search.jsp?R=20100036670
Kaye, R. (2010). *HE 75 Human factors engineering - Design of Medical Devices.* Rockville MD: U. S. Food and Drug Administration.
Kaye, Ron,. (2010). *HE75 Human factors engineering - Design of medical Devices.* Rockville MD: FDA.
Lano, R. (1977). The N2 Chart. . *TRW Software Series, Redondo Beach, CA*.
Microsoft. (n.d.). *Description of the database normalization basics* . Retrieved from Microsoft Support: https://support.microsoft.com/en-us/kb/283878
NASA. (2007). *NASA Systems Engineering Handbook NASA/SP-2007-6105 Rev1.* Washington DC: NASA.
Penn State University College of Health and Human Development. (2015). *Introduction to Factorial Experimental Designs*. Retrieved from The Methodology Center.
Phadke, M. S. (n.d.). *Introduction to Robust Design (Taguchi Method)*. Retrieved from iSixSigma: http://www.isixsigma.com/methodology/robust-design-taguchi-method/introduction-robust-design-taguchi-method/
Pugh, G. P. (1981). *Introduction to System Dynamics Modeling with Dynamo .* Pegasus.
Pugh, G. P. (1981). *Introduction to System Dynamics with Dynamo .* Pegasus Communications.
Tague, N. R. (2004). QThe Quality Toolbox. In N. R. Tague. Retrieved from http://asq.org/learn-about-quality/idea-creation-tools/overview/affinity.html

U.S Food and Drug Administration. (2015, April). *CFR - Code of Federal Regulations Title 21.* Retrieved from FDA: http://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfcfr/CFRSearch.cfm?CFRPart=820

U.S. Department of Health and Human Services, Food and Drug Adminstration. (2011, June 22). Draft Guidance for Industry and Food and Drug Adminstation Staff. *Applying Human Factors to Usabilty Engineering to Optimize Medical Device Design*. Rockville, Maryland, USA: U.S. Department of Health and Human Services, Food and Drug Adminstration.

Vanek, C. (2012, April 24). *Likert Scale – What is it? When to Use it? How to Analyze it?* Retrieved from SurveyGizmo: http://www.surveygizmo.com/survey-blog/likert-scale-what-is-it-how-to-analyze-it-and-when-to-use-it/

Wilson, B. (n.d.). *Five-by-five Whys*. Retrieved from The Rootisserie: http://www.bill-wilson.net/b73

