State of Tennessee

Division of TennCare

TennCare Requirements Management Standard

Version: 1.0
September 2020

TABLE OF CONTENTS

TABLE OF TABLES

TABLE OF FIGURES

Table 1: Requirements Management Standard Version History

Revision	Description of Change	Author	Date
1.0	Approved by TARB.	KPMG	September 17, 2020

1. Executive Summary

Requirements management is the capability to plan, execute, monitor, and control the development and implementation of business and system requirements. This Requirements Management Standard governs this capability within a project. Requirements management is important because requirements represent the business needs of an organization that must be accomplished for the successful completion of a project and to achieve a business goal or outcome, and support certification validation if applicable.

The Requirements Management Standard assumes that requirements management starts at the beginning of a project and governs the management of requirements throughout the project lifecycle.

If a project wants to deviate from this standard, the proposed Requirements Management Plan for that project must document the required changes and be submitted for approval by TennCare. The Project Steering Committee and the TennCare Architecture Review Board (TARB) will decide on any deviations from the standard.

Requirement Types, Attributes, Lifecycle, and Roles & Responsibilities

TennCare projects will define requirements using the Business Analysis Body of Knowledge (BABOK™), which uses the hierarchy of business and solution (functional and non-functional) requirements. The Requirements Management Standard defines the attributes which should be described as part of defining a requirement. Stakeholders only need to record the minimum essential information for requirements, so that the process is not cumbersome and therefore not followed. The Requirements Management Standard documents a simple requirements lifecycle, from gathering requirements through to implementation. The lifecycle processes incorporate all stakeholders in the Roles and Responsibilities section.

This standard recommends that the best practice for the development and implementation of requirements is for TennCare, or a vendor Partner on the behalf of TennCare, to manage the gathering and approval of solution requirements. Solution requirements are published in procurement documents such as Requests for Proposal (RFPs), with the intention of contracting a solution vendor to implement a solution. The level of detail elaborated for the purposes of procurement are very high-level and generally not written in sufficient detail to be testable. These procurement-level solution requirements are sufficient for vendors to bid on RFPs but are not written at a level of detail required to build and test a solution. New requirements can be uncovered at any time during a project and must be managed throughout their lifecycle.

Once a solution vendor is awarded a contract, the vendor must elaborate the contractual solution requirements into more detailed project solution requirements that are used in the design, implementation, and testing of the solution. All requirements should be testable after the vendor elaborates the requirements. For Commercial Off-the-Shelf (COTS) products and Solution as a

Service (SaaS), a comparison between the solution and the contract requirements is necessary to identify any gaps in the solution. Any identified gaps will be addressed by extending the solution, cancelling requirements, or additional configuration.

Roles and responsibilities are defined in a matrix mapping key activities against the various stakeholders involved in the requirements lifecycle.

Reporting and Quality Measurement

The Requirements Management Standard also defines how requirements are published, reported upon during their lifecycle, and how the quality of requirements will be tracked and measured. Requirement quality must be measured throughout the entire lifecycle.

Continual measurement of requirements volatility is a key factor in ensuring project success. The cost associated with requirements rework and poor definition results in substantial project cost overruns, delays, and scope creep.

Requirement Tools, Repository, and Traceability

This Requirements Management Standard includes a standard for all TennCare system projects to use to ensure traceability of business objectives through to implemented solutions via requirements management. The Requirements Management Standard also describes and defines TennCare standards for interoperability of tools and the use of requirements repository tools.

COTS solutions or SaaS may not have a Design, Develop, and Implement (DDI) phase, or may be truncated. However, there will be requirements that must be elaborated to a detailed enough level so that they can be tested.

This process assumes that TennCare IS projects are implementing iterative or waterfall style development projects. If an agile or other software development methodology is used, such as proof of concept phases, the solution vendor and TennCare will agree to the TennCare Solution Implementation Lifecycle (SILC) methodology used through Deliverable Expectation Documents (DEDs) during vendor onboarding.

Solutions vendors may have their own requirements management tools and methodologies that need to be aligned with the Requirements Management Standard guidance provided in this document. These tools may not be able to accommodate all elements of the requirements management lifecycle described in this document (e.g., the requirements traceability to architectural models). Deviations between a vendor’s toolset and the TennCare IS Software Whitelist must be reviewed by the TARB.

2. Requirements Management

2.1. Requirements Management and Solution Design

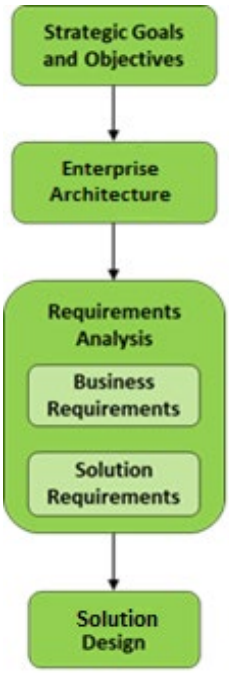

Figure 1: Requirements Management Process

The process of designing a solution and the definition and management of requirements are closely linked within the practice of business and systems analysis. This document only defines the framework for managing requirements - the practice of business analysis and solution design are highly related but not the same.

To define the scope of requirements management within this context, it is important to understand that:

- Requirements are focused on the business need; designs are focused on the solution
- The distinction between requirements and designs are not always clear
- The same techniques are used to elicit, model, and analyze both requirements and design
- A requirement leads to a design, which may drive the discovery and analysis of more requirements

Example differences between requirements and design are listed in *Table 1: Examples of Requirements versus Design*:

Table 2: Examples of Requirements versus Design

Requirement	Design
View six months sales data across multiple organizational units in a single view	A sketch of a dashboard
Reduce amount of time required to determine eligibility of an applicant	Process model
Record and access a medical patient's history	Screen mock-up showing specific data fields

2.2. Requirements Types

This standard uses definitions from the International Institute of Business Analysis (IIBA) – Business Analysis Body of Knowledge (BABOK®) Version 3.0, adapted for TennCare use.

A **requirement** is defined as a usable and testable representation of a need. Requirements focus on understanding what kind of value will be delivered if a requirement is fulfilled. The nature of the representation may be a document (or set of documents) but can vary widely depending on the circumstances. The BABOK® guide classifies requirements in the following way:

Business requirements are defined as statements of goals, objectives, and outcomes that describe why a change has been initiated. They can apply to the whole of an enterprise, a business area, or a specific initiative. They answer the question, *'Why do I need this?'*

Solution requirements are defined as the capabilities and qualities of a solution that meet the business requirements. They provide the appropriate level of detail to allow for the development and implementation of the solution.

The solution requirements are described at two levels specific to TennCare: **contractual** and **project**. Contractual solution requirements describe at a high-level what requirements are needed by the business, at a level of detail sufficient for procurement. They answer the question, *'What do I need?'* For example:

"The solution shall affirm and authenticate the consumer's identity through a prescribed and secure validation process."

The above requirement statement is high-level but contains sufficient information that a vendor can propose a solution to meet this requirement and estimate costs. This statement, however, is not written at sufficient detail to be testable.

The contractual requirements are elaborated into project requirements to make them testable:

1. The Member Portal shall utilize the FDSH RIDP service to validate the head of household's identity as part of the application process.
2. The solution shall ask an applicant three (3) "out of wallet" questions for authentication.
3. The solution shall ask two (2) supplemental questions when one authentication question is answered incorrectly.
4. The solution shall quarantine the email address and Social Security Number (SSN) of a user that fails the authentication process.
5. The solution shall require an applicant to complete an in-person process to be completed before the application can be released from quarantine.

Contractual and project solution requirements are divided into two sub-categories:

Functional requirements describe the capabilities that a solution must have in terms of the behavior and information that the solution will manage: *"The solution shall submit payment to a provider upon approval of an invoice."*

Non-functional requirements, or quality of service requirements, do not relate directly to the behavior of functionality of the solution but rather describe conditions under which a solution must remain effective or qualities that a solution must have: *"The solution shall support 10,000 concurrent users."*

See Figure 2 for a model of the requirement types.

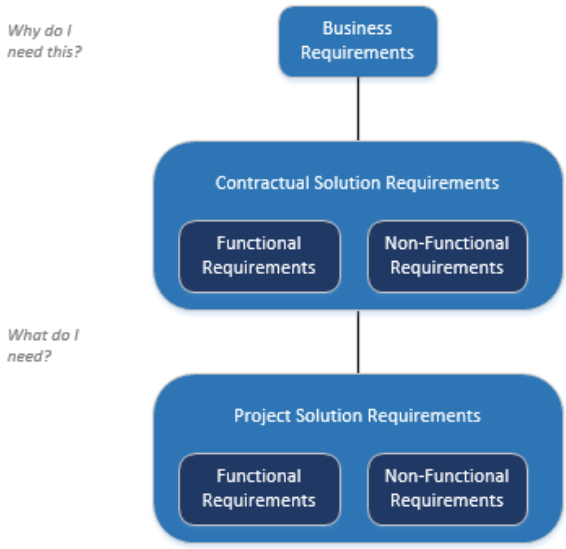

Figure 2: Requirement Types

2.3. Relationship Between Requirements and Deliverables

Figure 3: Relationship between Requirements and Deliverables illustrates the relationship between the different types of requirements and the documents or deliverables within which they are published. This is a generic pattern, which may need to be adapted depending on the needs of a specific TennCare IS project.

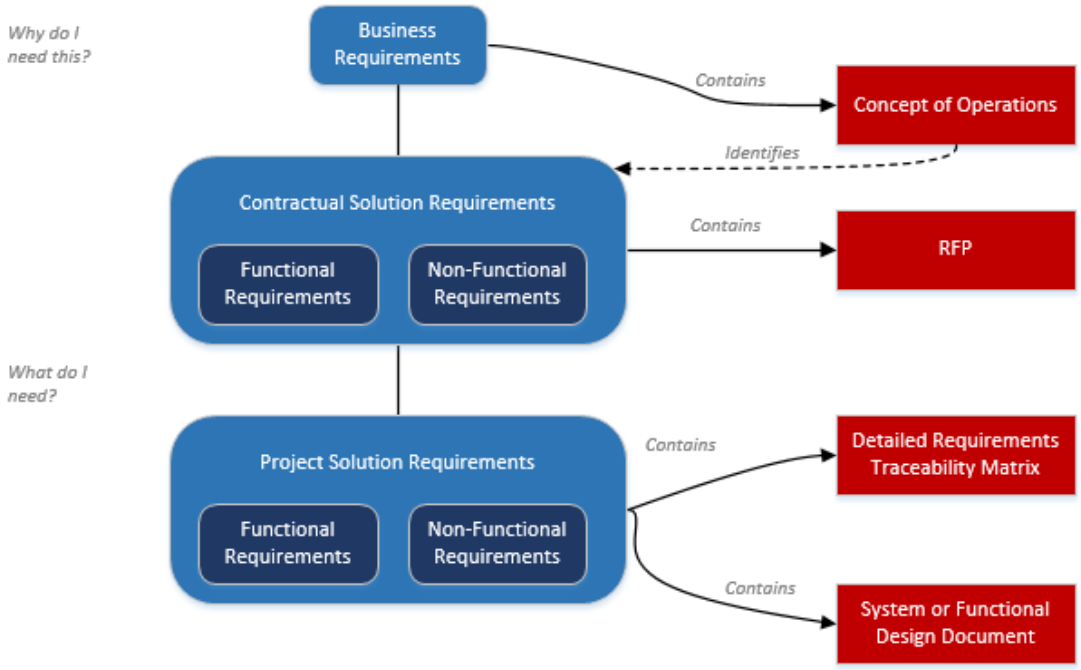

Figure 3: Relationship between Requirements and Deliverables

TennCare describes business requirements, business process models, context models, stakeholder analysis, information models, and other architectural artifacts to develop a solution's Conceptual Design. The business requirements and other architectural models published in the Conceptual Design are used to define the contractual solution requirements, which are included in RFPs.

Once a solution vendor is contracted, they must elaborate the contractual solution requirements (both functional and non-functional) into more detailed project-level solution requirements that can be tested. These requirements can be used to develop a more detailed Requirements Traceability Matrix (RTM). Solution requirements are usually published with other types of requirement artifacts, such as use cases and logical data models, in a System or Functional Design Document.

2.4. Requirements Traceability Matrix

The RTM provides traceability from contract requirements to project requirements, from project requirements to design elements, and design elements to test cases and defects. The RTM is vital to ensuring all requirements are tested during the appropriate testing stage. The requirements in the RTM must continue to align to TennCare's target state enterprise architecture and requirements as published in the RFP for a solution vendor. Once the solution vendor is on-boarded, requirements are elaborated to a level of detail that is testable, with defined acceptance criteria. The RTM approved at the Requirements Review Gate must have project-level solution requirements traced to the original contract requirements and to any Centers for Medicare and Medicaid Services (CMS) compliance criteria that are required for certification activities.

The RTM is normally a matrix with one row per elaborated project requirement. *Table 2: Requirements Traceability Attributes* recommends the columns for TennCare solution vendors when developing the RTM. Projects may modify and add columns as necessary or at the discretion of TennCare.

Table 3: Requirements Traceability Attributes

Contractual Solution Requirements	
Business Area	The organization unit of TennCare that needs this requirement.
Business Function	The MITA Business Function supported by this requirement.
Conceptual Design Element	Reference to a business process, data entity, or other element of the Conceptual Design that articulates the requirement in the target architecture.
RFP Requirement ID	The unique identifier for this requirement during the RFP.
RFP Requirement Text	The name and description for this requirement during the RFP.
Project Solution Requirements	
Requirement ID	The unique identifier for this updated requirement after the RFP.
Requirement Text	The name and description for this updated requirement after the RFP.
Comments	Additional text for notes, comments, and details about the updated requirement.
Status	The status of this requirement. Options will include: Drafted, Approved, Implemented, Verified, Deferred, Deleted, Rejected, and any other relevant fields
Test Cases, Scenarios, and Scripts	
Test ID	The unique identifier for this test case, scenario or script.
Testing Stage	The stage of testing when this test case, scenario, or script will be tested. The list of test stages includes: Unit Testing, System Integration Testing, User Acceptance Testing, Operational Readiness Testing, and Beta testing Stage. More information available in the TennCare Test Management Standard.
Test Status	The status of this test case, scenario or script. Options should include: Waiting to be Tested, Being Tested, Deferred Testing, Failed Testing, Passed Testing with exemption, Passed Testing, etc.
Application Component	

Module Name	The application module that satisfies the requirement. As defined in the Application Component Model or Solution Architecture Model (see the EA Modeling Standard).
Application Element	
Name or identifier	The part of an Application Component that satisfies a Requirement. E.g. an API, widget, UI element, callable service, module entry-point, etc. Sufficiently precise to locate a defect.

The Requirement Traceability Semantic Model in Appendix D details how the project solution requirement and the application component are mapped together and how the project solution requirements are then tested through a test case.

2.5. Requirements Lifecycle

Error! Reference source not found. Figure 4 describes the states a requirement goes through during its lifecycle, specifically when a solution vendor is engaged to design, develop, and implement a solution.

Once a requirement is baselined, it is subject to the requirements change management process, as a subset of the project change management process.

If a solution is a source-code transfer, a COTS solution, or a SaaS, a walkthrough of the solution components to requirements is performed in the project's Requirements Review Phase.

Once development is complete, the solution components related to the requirements are tested and implemented. A requirement will then be updated to have the status of implemented. Along its lifecycle, a requirement can be canceled or deferred to a different release. At any time, new requirements can be identified and must be managed through the entire lifecycle.

This model specifically describes the statuses of contractual and project solution requirements. Business requirements do not have the status of being verified because they are at a higher level of granularity.

The RACI matrix presented in the Requirements Management Roles & Responsibilities Section maps the roles that can move requirements through their lifecycle.

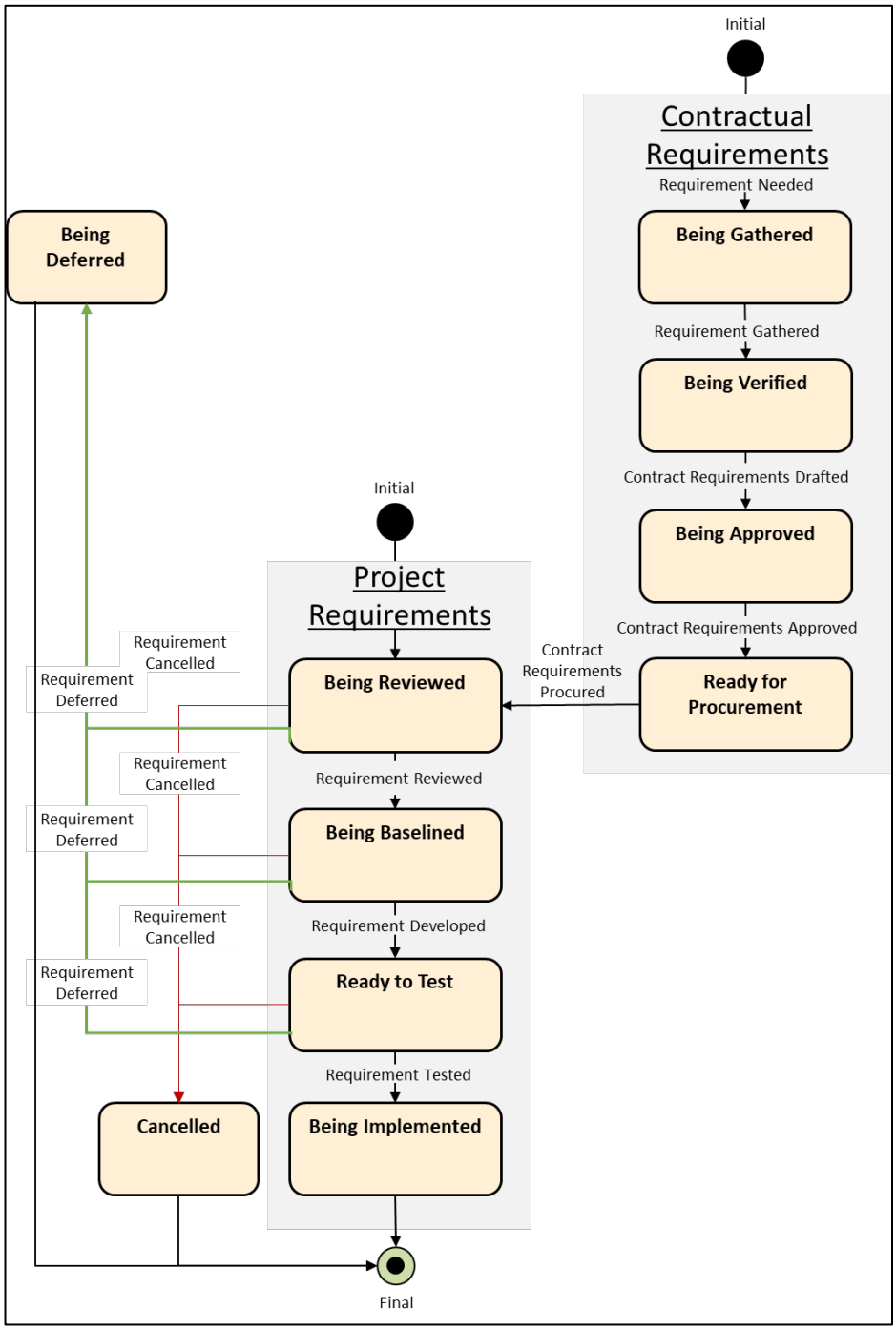

Figure 4: Requirements State Transition Model

Requirements elaboration and traceability will occur through all phases of a project.

In the Planning phase, contractual requirements are developed and articulated by TennCare, or by a vendor Partner, describing high-level solution functional and non-functional requirements ('contractual').

After a solution vendor is contracted, they will elaborate the contractual requirements to a testable level during the Requirements Review and Elaboration phase. Through the Design and Testing phases, the requirements will be traced against design artifacts, test cases, and defects. Figure 5 aligns all the phases, their respective activities, and the traceability between the requirements.

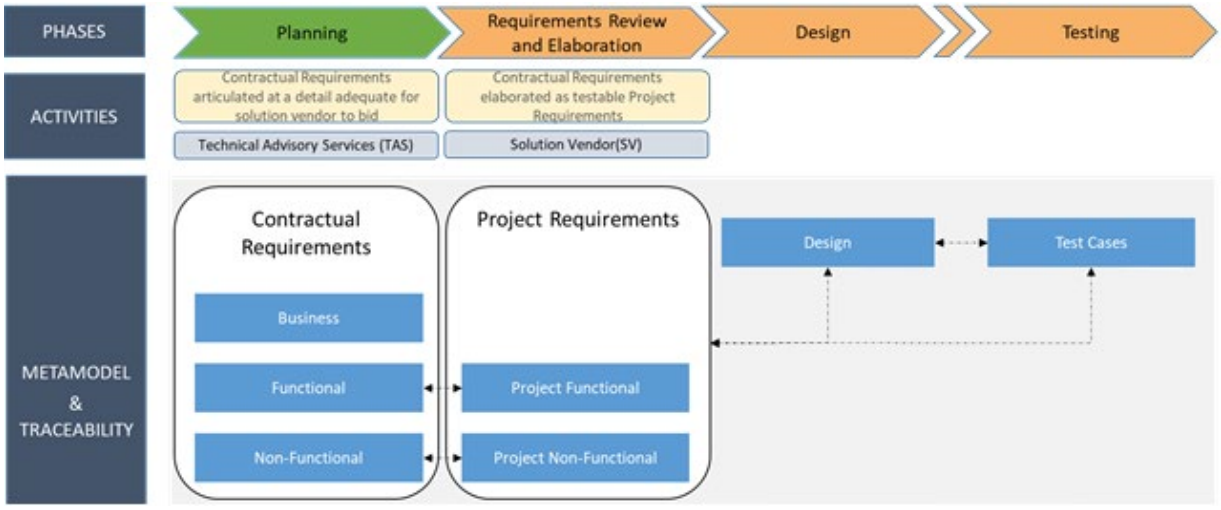

Figure 5: Requirements Elaboration and Traceability

Prioritization activities will occur in the Requirements Review Phase to rank requirements and determine their relative importance to stakeholders. Prioritization should occur as the contractual requirements are being elaborated. Prioritization may change based on contractual obligations. All contractual requirements must be met by the solution vendor unless the contract is amended. Project requirements can be prioritized after contract signing, or can also be waived at the approval of TennCare, as long as waiving the project requirement does not put fulfillment of the contractual requirement at risk.

The following criteria should be used to determine a requirement's priority:

- **Regulatory or policy compliance:** requirements that must be implemented to meet regulatory or policy demands
- **Benefit:** the advantage implementing the requirement will accrue to stakeholders
- **Penalty:** the consequences resulting from not implementing the requirement
- **Cost:** the effort and resources needed to implement the requirement
- **Risk:** the chance that the requirement cannot deliver its potential value
- **Dependencies:** relationships between requirements where one is dependent on another requirement being implemented

- **Time sensitivity:** how soon the requirement will need to be implemented before it loses significant value
- **Stability:** the likelihood the requirement will change, either because it requires further analysis or because stakeholders have not reached a consensus

Each of these criteria should be assigned a weight relative to their importance in the project and the organization.

2.6. Requirements Attributes

Both *Figure 6: Solution Requirement Attributes* and *Table 3: Solution Requirement Attributes* illustrates the attributes for contractual and project-level solution requirements. Business requirements may not require all of these attributes.

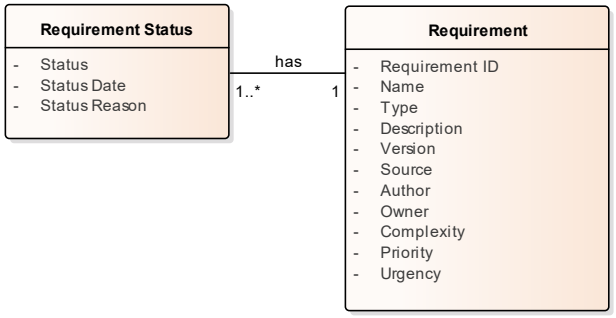

Figure 6: Solution Requirement Attributes

Table 4: Solution Requirement Attributes

Attribute	Description
Requirement ID	Unique reference that is not altered or reused if the requirement is changed.
Name	Requirement title or short name.
Type	Identifies the type of requirement (e.g. business, contractual, or project functional and non-functional). For non-functional requirements, the follow sub-types are suggested: 1. Privacy 2. Security 3. Usability 4. User Interface 5. System Interfaces 6. Accessibility 7. Business Environment

Attribute	Description
	8. Information Security 9. User Access 10. Performance/Capacity 11. Business Continuity/Recoverability 12. Logging/Monitoring 13. Archive/Retention 14. Expected Life Span 15. Documentation/Training/Help 16. Communication 17. Legislative and Regulatory/Compliance 18. Integration 19. Data Migration 20. Deployment
Description	Gives a detailed description of the requirement. The level of detail will vary depending on the requirement type. Project solution requirements will be at a lower level of detail than contractual solution or business requirements. A contractual solution requirement will elaborate into many project requirements.
Version	Requirement increment, along with changelog and historical documentation.
Source	Identifies the origin of the requirement. The source is often consulted if the requirement changes or if more information regarding the requirement or the need that drove the requirement has to be obtained.
Author	Provides the name of the person who needs to be consulted should the requirement later be found to be ambiguous, unclear, or in conflict.
Owner	Indicates the individual or group that needs the requirement or the business owner after the solution is implemented.
Complexity	Indicates how difficult the requirement will be to implement. This will assist in estimating how long a development effort is required, or the release to which it will be allocated. Complexity is measured as: 1. This requirement is not very complex a. The requirement does not require a large amount of effort to implement. b. This requirement does not require input from a Subject Matter Expert to implement. c. This requirement does not require integration with outside systems or interfaces.

Attribute	Description
	d. This requirement does not require a large amount of change to the implementation or organization. e. This requirement does not have a large security impact or risk profile. 2. This requirement is somewhat complex a. This requirement requires more than a small amount of effort to implement. b. This requirement requires consultation with a Subject Matter Expert during implementation. c. This requirement requires implementation with outside systems and services that are already in a steady operational state. d. This requirement will require minor changes to the organization. e. This requirement has some security impact and risk profile, but it is thoroughly managed. 3. This requirement is very complex. a. This requirement requires a large amount of effort to implement. b. This requirement requires a Subject Matter Expert to implement. c. This requirement has a major security impact or risk profile, and without proper management can cause major impact to the project or organization. d. This requirement requires implementation with outside systems and services that are not yet in a steady operational state.
Priority	Indicates relative importance. Priority can refer to the relative value of a requirement or to the sequence in which it will be implemented. 1. High a. Implementing this requirement is imperative for the overall solution. b. This requirement must be implemented along with the solution, and if it is not, it will be a major block to the solution. c. This is a high risk requirement and should be addressed sooner rather than later. d. This requirement's completion is a dependency on other high priority requirements. e. This requirement has a set timeline to implement. f. This requirement requires additional staff focus to implement. g. This requirement is required to meet mandatory regulatory or policy demands. 2. Medium a. Implementing this requirement is needed for the overall solution.

Attribute	Description
	b. This requirement needs to be implemented along with the solution, but will not stop the overall solution if the requirement needs to be adjusted. c. This requirement has some risk, but the risk can be easily managed or mitigated in order to successfully implement the requirement. d. This requirement's implementation does not have dependencies on other high priority requirements. Any dependencies on medium requirements can be mitigated with workarounds if this requirement is not implemented. e. This requirement has a set timeline, but that timeline can be adjusted without introducing risk to the overall solution. f. This requirement does address some regulatory or compliance demands, but they can be met through other requirements or workarounds. g. This requirement will likely not change, however there is still further analysis or stakeholder review required. 3. Low a. Implementing this requirement is helpful in the overall solution, but is not crucial. b. This requirement can be implemented at a later time without major impact to the overall solution. c. This requirement has little to no risk associated with its implementation. d. This requirement's implementation has no dependencies on other high or medium priority requirements. e. This requirement does not have a set timeline. f. This requirement has a high likelihood of being changed. g. This requirement is not required to meet regulatory or policy demands.
Urgency	Indicates how soon the requirement is needed. It is usually only necessary to specify this separately from the priority when a deadline exists for implementation.
Status	Indicates the state of the requirement, as per the allowed requirement states.
Status Date	The date on which the status became effective.
Status Reason	The reason that the status was set – for example, if the status is canceled or deferred.

Each requirement must also have a set of traceability relationships. These are not captured as attributes but as a set of relationships to:

- Other requirements to show dependence or sequence
- Other requirements to show groupings (such as a hierarchy of functionality)
- Other requirements to show traceability to parent or child requirements (e.g. traceability from parent business requirements to child contractual solution requirements, or from parent contractual solution requirements to project solution requirements)

It will also be important to track relationships to other elements such as:

Activities – solution requirements have relationships to activities on process models.

Application Components – solution requirements have relationships to the application components that satisfy that requirement.

Business Rules – solution requirements have relationships to business rules which are invoked in the requirement.

Regulatory Requirements – depending on the project, the requirements may need to be traced back to regulatory requirements that the project requirements are fulfilling (e.g. CMS requirements).

Releases – the current and past release allocation history for the requirement. It is also important to know the release status, for example, whether the release is built, tested, deployed, etc. Once the release that the requirement is allocated to is implemented, then the status of the requirement should be set to be implemented.

Stakeholders – to show ownership of requirements.

Test Cases – the set of test cases, and their test statuses, associated with the requirement. It is important to know if these tests are completed, passed, or have defects which are being managed. Defects are related to test cases.

Appendix D, Requirement Traceability Semantic Model**,** contains a model describing the relationships between requirement elements and other design or architecture elements.

2.7. Requirements Quality and Writing Style Guide

In general, good quality requirements are:

- **Complete** – Fully describes the functionality to be delivered, how that functionality will be measured, or the need of the business.
- **Consistent** – Requirements should not conflict with other requirements of the same type or with higher-level requirements.

- **Correct** – Accurately describes the functionality to be built.
- **Feasible** – Can be implemented within the known capacities and limitations of the system and its operating environment.
- **Necessary** – Documents a needed capability or one required to comply with an external system requirement or a standard.
- **Prioritized** – Is assigned an implementation priority (in the case of each functional requirement) to indicate how essential it is.
- **Unambiguous** – Enables all readers of a requirement statement to arrive at a single, consistent interpretation of it.
- **Verifiable** – Indicates through tests or other verification approaches, such as inspection or demonstration, that it implements the required action properly.
- **Traceable** – Can be linked backward to its origin and forward to the design elements and source code that implement it as well as to the test cases that verify it was implemented correctly.
- **Testable** – the requirements must be able to be tested in the solution.

There are a few writing style guidelines that should be followed to create requirements that are easy to understand and implement:

- **Clear and concise:** write requirements that are easy to understand. This means keeping the requirements short and direct, and written in complete sentences using proper grammar, spelling, and punctuation. Avoid jargon where possible; ensure the requirements can be understood by the majority of readers. Write concisely; phrases like *"needs to provide the user with the capability to"* can be condensed to *"shall".* Regardless of the terminology used, keep it consistent across all requirements. If a set of information is not valuable, consider removing it.
- **Active voice:** write requirements in the active voice rather than the passive – make it clear *who* will be taking the action described. *"Upon payment of a claim, the claim paid date line will be updated"* is written so the performer of the action is unclear. Writing the requirement as *"The system shall update the claim payment data line with the date the claim was paid when the claims payment manager confirms receipt of invoice claim payment"* gives a clear indication as to who/what perform each activity.
- **Individual Requirements:** write requirements to be individualized; avoid long narratives that contain multiple requirements. Try to avoid using the words "and", "or", "additionally", or "also" – these words often facilitate a combination of multiple requirements. Also avoid using "unless", "except", or "but" as they facilitate and encourage the use of multiple requirements. Instead of writing: *"The client's credit card on file shall be charged for payment, unless the credit card has expired"*, split them into two requirements: *"If the client's credit card on file is active, the system shall charge the payment to that card"* and *"If the client's credit card on file has expired, the system shall allow the buyer to either update the current credit card information or enter a new credit card for payment."*

Requirements need to be specified at a level of detail that provides developers and testers with just enough information to properly implement them:

- **Appropriate detail:** provide enough detail to minimize the risk of any misunderstandings to occur for the development team and other stakeholders. The fewer opportunities for communication with the development team about requirements issues, the more detailed the requirements need to be. The developer should be able to think of several *acceptable* ways to satisfy a requirement.
- **Consistent granularity:** write functional requirements within a set of related requirements at a consistent level. Write individually testable requirements. If there are a small number of related test cases to verify a requirement was correctly implemented, it's likely at an appropriate level of granularity. On the other hand, there are a wide variety of tests for a single requirement, the requirement may not be fully elaborated. *"The system shall interpret the keystroke combination Ctrl + S as File Save"* is at an appropriate level of granularity, as it will require few tests to verify. A requirement like *"the system shall respond to editing directives entered by voice"*¸ on the other hand, would require hundreds of tests to verify due to the number of combinations.

The way requirements are represented and communicated play a large role as to how they are read and interpreted. Consider a set of requirements of a similar pattern with multiple combinations: *"The text editor shall be able to parse* ***<format>*** *documents that define* ***<jurisdiction>*** *laws"*. That requirement format can have, for example, 12 different requirements to read that are quite similar (3 possible values for **<format>**, and 4 for **<jurisdiction>**). This can lead to having requirements that are duplicated or missing. To avoid that, and make them easier to read, have the values in a table/matrix that can be referenced by requirements: *"The text editor shall be able to parse documents in several formats that define laws in the jurisdictions as shown in Table 11.1."*

Write requirements to be clear and free from any ambiguities or problems – these lead to unverifiable requirements. The best way to ensure a requirement is clear and to avoid writer's bias is to have them peer reviewed. These techniques can assist:

- **Avoid fuzzy words:** have a defined business glossary, and ensure those terms are agreed upon and used consistently; if there are multiple terms for something, agree on one.
- **Avoid adverbs:** they introduce subjectivity and ambiguity. Avoid words like *reasonably, appropriately, generally, approximately, usually, systematically,* and *quickly*. In general, use words that are quantifiably measurable.
- **Avoid the A/B construct:** combining two related, synonymous, or opposite terms with a slash frequently lead to ambiguity, and have many different ways of interpretation by the reader. For example, the requirement *"The system shall provide automated information collection of license key data for a mass release from the Delivery/Fulfillment team"* can be interpreted to mean the name of the team is Delivery/Fulfillment, delivery and fulfillment are synonyms, either the Delivery or the Fulfillment team do a mass release, or both the Delivery and the Fulfillment teams jointly do a mass release.
- **Clarify boundary values:** ambiguities can occur at the boundaries of ranges. The following requirement "Vacation requests of up to 5 days do not require approval. Vacation requests of 5 to 10 days require supervisor approval. Vacation requests of 10 days or longer require

management approval." It is unclear exactly which category vacation requests of exactly 5 days or 10 days belong. Make it clear in the requirements whether the endpoints lie inside or outside the range. The words through, inclusive, and exclusive should help make the endpoints clear: "Vacation requests of 5 or fewer days do not require approval. Vacation requests of longer than 5 days through 10 days require supervisor approval. Vacation requests of longer than 10 days require management approval."

- **Write positive requirements:** write requirements in a way that state what the system or user will do rather than what they will not do. Writing in a negative can lead to double or triple negatives that are more difficult to decipher. Rephrase negative requirements, such as *"Prevent the user from activating the account if the account is not in balance."* to *"The system shall allow the user to activate the account only if the account is in balance."* This avoids potential inconsistencies in a reader misinterpreting the requirements due to a complicated chain of negatives.
- **Ambiguous phrases:** Avoid adding ambiguous phrases, such as "etc.", "including but not limited to", and "and others", which can lead to scope creep.

Although there is no absolute way to avoid missing or incomplete requirements, you can mitigate the occurrence by avoiding writing requirements in the following ways:

- **Symmetrical operations:** due to their nature, symmetrical operations may not always gather all necessary requirements to make it clear to the developer. As an example, the requirement: *"The user must be able to save the account at any point during manual account setup"* opens a whole host of questions. Can a user retrieve and work on an incomplete but saved work? Should the system validate data entries in incomplete contracts before saving?
- **Complex logic:** compound logical expressions can lead to undefined decision values. In the following example "If the premium plan is not selected and proof of insurance is not provided, the customer should automatically default into the basic plan." Although this requirement refers to two binary decisions, the combinations lead to four possible outcomes. This requirement only addresses one combination however, and the developer may conclude that no action is taken for the three other conditions.
- **Missing exceptions:** systems should provide requirements as necessary to describe how it will respond to exceptions. Ensure that the exception requirements have accompanying requirements that address all potential exceptions. The following requirements: *"If the user is working in an existing file and chooses to save the file, the system shall save it with the same name"* does not indicate what the system should do if it's unable to save the file with the same name. Ensure the above requirement is accompanied by: *"If the system is unable to save a file using a specific name, the system shall give the user the option to save it with a different name or to cancel the save operation."*

Checklists are provided in Appendix C which can be used and adapted to verify requirement completeness and quality.

2.8. Requirement Review, Approval, and Alignment to TennCare SILC

Business requirements are developed during a project's Concept Phase. Business requirements are used to develop a solution's Conceptual Design, i.e. its target state architecture. This is approved by the Conceptual Design Review, part of the Architecture Review (AR) gate. Business requirements are high-level statements of the abilities that a business needs to achieve by implementing the project (e.g. the business needs the ability to process and adjudicate Medicaid claims).

Contractual functional/non-functional solution requirements are developed in the Planning Phase. These contractual solution requirements are used to prepare an RFP for the Project Baseline Review.

Once a solution vendor is contracted, contractual solution requirements are elaborated to traceable and testable project solution requirements during the Requirements Review Phase. The elaborated requirements are used to develop a detailed RTM, and are reviewed as part of the Requirements Review gate.

In the Design Phase, the project solution requirements are used to develop the Technical and Functional Design Documents reviewed as part of the Final Detailed Design Review gate.

Project solution requirements are traced against the contractual solution requirements as they are being developed. In the Design Phase, the elaborated project solution requirements are traced against design artifacts to ensure they are being met. Through the Testing Phase, project solution requirements are traced against test cases to ensure the solution is meeting all requirements. Once implemented, the project lifecycle begins again for post-implementation activities.

The diagram attached in Appendix D illustrates a traceability model for the requirement and architecture elements. Traceability of requirements to other design objects, or to other requirements, are captured in the relationships that the requirements have to these objects and in the requirement attributes. Figure 7 does not specify all of the relationships between the architecture and requirement elements but does specify those that are required for traceability.

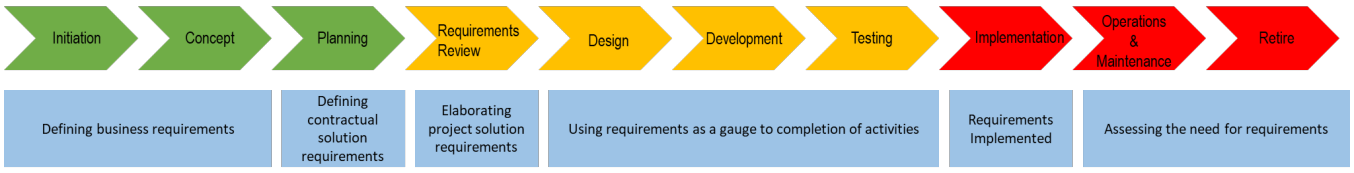

Figure 7: Requirements in a Project SILC

At a minimum, all requirements must meet the following milestones:

- Business owners validate, review, and approve the contractual requirements for their functional areas, or amend them as appropriate, during the Planning Phase.
- The Requirements Management Plan for the project is aligned with the activities and artifacts within the TennCare SILC gate reviews. This plan is produced as the vendor is on-boarded at the start of the Requirements Review Phase.
- TennCare and the solution vendor review contractual requirements against a solution design or proposed solution. The solution vendor elaborates each contractual solution requirement into testable project requirements or new project requirements. Once approved, project requirements are then baselined. This occurs during the Requirements Review Phase.
- The solution vendor provides a design or proposed solution that aligns with the baselined project requirements for TennCare approval during the Design Phase.
- The solution vendor builds the approved design based on baselined project requirements or configures a SaaS during the Development Phase.
- The solution vendor executes test cases to demonstrate how the project requirements are met during the Testing Phase.
- The solution vendor implements the approved design, based on the approved project requirements, in the production environment during the Implementation Phase.

3. Requirements Management Roles & Responsibilities

The following roles are defined for stakeholders in the RACI charts within this standard. The RACI acronym means:

(R) Responsible: Those who are primary responsible for the work to complete the task or deliverable. Only one party shall be responsible for any activity, task, or deliverable.

(A) Accountable: Those who are accountable for ensuring the correct and thorough completion of the task or deliverable. There should be only one Accountable party for any activity, task, or deliverable.

(C) Consulted: Those whose opinions and input are sought (two-way conversation).

(I) Informed: Those who are kept up to date on progress, often only on completion of the task or deliverable (one-way conversation).

In the RACI chart, where the activity is to “approve” a deliverable, the A indicates who approves it, and the R indicates who requests the approval.

Table 5: RACI Participants Definition

Stakeholder	Definition
Business Support Services	Assists with Strategic Planning and Development of Business Enterprise Architecture strategy; providing subject-matter expertise supporting certification, testing, Medicaid Information Technology Architecture (MITA), and Advanced Planning Documents; assistance preparing for and coordinating project reviews; and, assistance with the documentation of existing processes and recommendations for process improvements.
TennCare Business	Organizational units that oversee the policies and operations of TennCare business functions, such as member services.
TennCare IS	Provides support for planning, design, implementation and operation of information technologies and methodologies.
Program and Project Management	TennCare assigned project managers and resources responsible for the project level management.
Solution Vendor	Responsible for the design, development, testing, implementation, and the operations and maintenance (O&M) of solutions.

Stakeholder	Definition
STS (Infrastructure)	Provides direction, planning, resources, execution, and coordination in managing the information solutions needs of the State of Tennessee. STS is a division within the Department of Finance & Administration and may operate on premise solutions for TennCare, or may advise on vendor procured cloud-based solutions.
TAS	Supports and advises on the TennCare IS Portfolio by offering Organizational Change Management and Training, Operations & Maintenance Planning, Lifecycle Advisory Services, Quality Management, Strategic Project Management, and Enterprise Architecture services.

Table 6: Requirements Roles and Responsibilities

Lifecycle Project Phase	Lifecycle Stage Gate Review	Requirement Activites	TennCare Business	TennCare IS	Program/Project Mgmt	STS (Infra)	Solution Vendor	TAS	Business Support Services
Planning	Project Baseline Review	Determine business requirements	A	C	C	I	-	R	C
		Determine functional and non-functional requirements	A	C	C	I	-	R	C
Requirements Review	Requirements Review	Develop Requirements Management Plan	C	A	C	I	R	C	C
		Review contractual requirements and identify which ones need to be elaborated.	C	A	C	I	R	C	C
		Elaborate the contractual requirements into project requirements (business, functional, non-functional).	C	A	C	I	R	C	C
		Review project requirements	C	A	C	I	R	C	C
		Develop Detailed Requirements Traceability Matrix	C	A	C	I	R	C	C
		Prioritize requirements	C	A	C	I	R	C	C
		Approve Vendor specific Requirements Management Plan	C	A	C	I	R	C	C
		Approve Detailed Requirements Traceability Matrix	C	A	C	I	R	C	C
Design	Final Detailed Design Review	Create traceability between project requirements against design artifacts and CMS checklists.	C	A	C	C	R	C	C
		Create traceability between project requirements against test cases.	C	A	C	C	R	C	C

TennCare Business (business owner) and **TennCare IS** (IT owner) are accountable for ensuring that the requirements capture the business or technology need properly. TennCare IS is also

responsible for ensuring that the impact of requirements on the organization as a whole are understood and are balanced against the needs of an individual project during the activity of determining functional and non-functional requirements.

4. Management Tools, Repository and Reporting

4.1. Requirements Tools and Repository for TennCare Systems Projects

This section describes the approach for how requirements and architecture tools will interoperate within projects, and between TennCare and a project.

In general, TennCare, or a vendor Partner on TennCare's behalf, will create business and contractual solution requirements in TennCare's requirements management tool, the Sparx Enterprise Architect (EA) requirements and design repository. Contractual requirements from prior projects may be reused as a baseline to create contractual requirements for the proposed project. It is expected that solution vendors will import business and contractual requirements into the solution vendor's toolset, elaborate them to the project level and manage those project requirements through their full lifecycle.

It is expected that a project will synchronize requirements into TennCare's architecture and requirements repository.

4.2. Tools in the Context of TennCare Systems Projects

Figure 8: Requirements Context for Solution Vendor represents TennCare's operating environment for its architecture and requirements repository and how it is expected to interact with vendor repositories during a project.

Once a vendor is contracted to implement a solution based on the RFP, the vendor must take the contract requirements and further elaborate those to a level that is detailed and testable either within TennCare's repository and toolset, or within their own requirements management tool. These elaborated 'project requirements' must be traceable back to the original contract requirements in TennCare's repository. The vendor is expected to periodically synchronize their requirements into TennCare's repository, so that TennCare has a complete and accurate picture of requirements as they are elaborated throughout the project lifecycle. This also allows TennCare to complete impact analysis back to contract requirements when change requests are assessed.

The vendor must also product a series of solution archtitecture artifacts, and again may chose to use TennCare's architecture tool, or their own. If they use their own, the vendor is expected to synchronize back to TennCare's repository.

As a vendor completes the design, development, and implementation phases of the project, they are expected to keep a series of design elements, test cases, test results, and defects in their test management tool. All elements must be traced back to their project and original contract requirements and synchronized to TennCare's repository.

Regardless of the approach, it is expected that vendors will synchronize all material developed back into TennCare's architecture and requirements repository, and also ensure traceability of all architecture artifacts, requirements, design elements, test cases, and defects back to contract requirements.

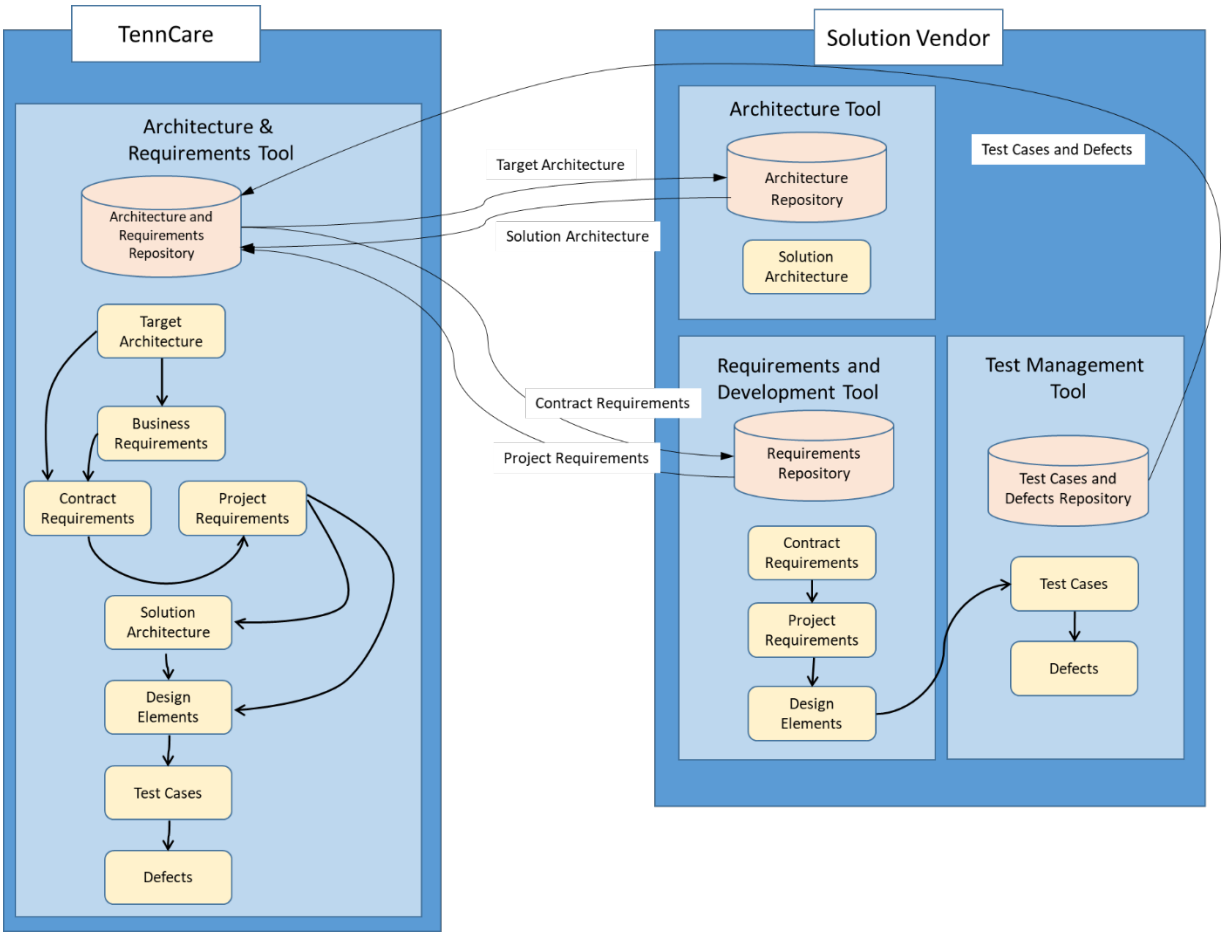

Figure 8: Requirements Context for Solution Vendor

4.3. Reporting

During projects, all stakeholders will require regular reports about the status of requirements, independent of formal publication of requirements documents. At a minimum, TennCare will be able to report the following, as required, on demand:

1. Requirements Stability Index
2. Detailed requirement report or RTM by:
   a. Requirement mapped to business rule or policy
   b. Requirement mapped to data element
   c. Requirement mapped to application component

d. Requirement mapped to owner
e. Requirement mapped to release
f. Requirement mapped to test case and defects

3. Total number of requirements:
   a. By timeframe, owner, and status
   b. Allocation by release
   c. Changed
   d. Cancelled
   e. Added
   f. Re-assigned to a later release
4. Which requirements are:
   a. Implemented in a release
   b. Still being tested
   c. Still in development
   d. Awaiting allocation to a release

4.4. Requirements Quality Measurement

This section describes how the quality of requirements will be measured and tracked throughout the requirements management lifecycle. This section contains the Key Performance Indicators (KPI) and metrics that can be used by TennCare to measure the quality of requirements.

In order to measure quality of requirements, the reviewers for the requirement related artifacts need to ensure that requirements are written appropriately. This means that requirements are written in clear and concise language with good quality. Requirements do not include the details of the design or the implementation. Determining the execution and implementation of a requirement is the responsibility of the solution vendor.

Once requirements are gathered and approved, a project must understand how requirements are impacting the project. This is accomplished through the collection of metrics and monitoring of the Requirements Stability Index KPI. Any such metrics collected must be demonstrative of a specific project problem that needs to be addressed once the acceptable threshold for that metric has been exceeded.

TennCare and vendors can work together to define how level of effort is described and measured in the Requirements Management Plan, if those KPI are used to measure requirement quality.

The measurements included in *Table 6: Requirement KPI Measurements* and *Table 7: Requirement Quality Metrics* represents a demonstrative sample of possible metrics that can be tracked to monitor the quality of requirements.

Table 7: Requirement KPI Measurements

KPI	Use	Project Impact	Additional Notes
Requirements Stability Index	Determines the overall rate of change to requirements. (Total # of Requirements Baselined + # of Changed Requirements + # of Added Requirements + # of Cancelled Requirements) / Total # of Requirements Baselined	If this rate of change is too high, requirements are potentially impacting schedule, attributing to scope creep, or are not being written properly.	This metric is best used for mapping volatility over time, identifying times when volatility was introduced into the requirements process. Correlating the volatility to activities that were occurring at the time will allow for adjustments to the process and activities to further avoid introducing high levels of volatility into requirements.

Table 8: Requirement Quality Metrics

Metric	Use	Project Impact	Additional Notes
Total number of baselined requirements changed	Can demonstrate a project-level problem with writing or defining requirements.	This will demonstrate if there is a large amount of requirements being changed on the project.	Used as part of determining the Requirements Stability Index.
Total number of baselined requirements canceled	Can demonstrate a project-level problem with writing and defining requirements.	This will demonstrate if there is a large number of requirements being cancelled on the project.	Used as part of determining the Requirements Stability Index.
Total number of requirements added after baselined	Can demonstrate a project-level problem with missing requirements.	This will demonstrate if there is a large number of requirements being added to the project.	Used as part of determining the Requirements Stability Index.
Number of baselined requirements being assigned to a later release	This movement can demonstrate schedule slippage.	If requirements are constantly not completed in their originally scheduled release, this can point to a problem with schedule slippage.	This metric can be more expeditious in its usefulness with Agile, projects where there are small sprints as part of the Work Breakdown Structure. For larger, Waterfall-style projects, this metric can help in planning schedule, scope, and budget for future releases.

Metric	Use	Project Impact	Additional Notes
Total number of baselined requirements changed by functional area	Can determine if one particular functional area (security, business, technical) or sub-function (user experience, services) are having difficulty writing acceptable requirements in a clear and concise manner with no ambiguity.	If one or more functional areas are struggling with the quality of their requirements, action plans and strategy can be directed toward that function or sub-function to ensure proper requirements are collected.	If one or more functions are not providing well written requirements, there can be downstream impacts on requirements in other functional or sub-functional areas that share dependencies on the requirements.
Total number of baselined requirements cancelled by functional area	Can determine if one particular functional area (security, business, technical) or sub-function (user experience, services) are having difficulty writing acceptable requirements in a clear and concise manner or identifying the appropriate requirements.	If one or more functional areas are struggling with the identification and quality of their requirements, action plans and strategy can be directed toward that function or sub-function to ensure proper requirements are collected.	If one or more functions are not providing well written requirements, there can be downstream impacts on requirements in other functional or sub-functional areas that share dependencies on the requirements.
Total number of requirements added by functional area since baselined	Can determine if one particular functional area (security, business, technical) is having difficulty identifying and defining acceptable requirements based on the characteristics of good requirements.	If one or more functional areas are struggling with the identification and definition of their requirements, action plans and strategy can be directed toward that function or sub-function to ensure proper requirements are collected.	Characteristics of good requirements (see **Error! Reference source not found.** Section): Written in clear and concise language with no ambiguity Testable Traceable Consistent Feasible Necessary Doesn't include implementation specifics

Metric	Use	Project Impact	Additional Notes
Level of effort of baselined requirements cancelled versus level of effort of requirements added since baselining	This can demonstrate the level of scope creep.	This demonstrates scope creep that may lead to additional Change Requests, or Scope, Budget, and Schedule discussions.	Some requirement management programs capture LOE based upon time, some use a rating or complexity level of small, medium, or high. In some cases this metric will be comparing total time vs. new total time – showing how much more or less effort has been changed. Other times the number of cancelled small, medium, and highs will be compared to the new number of small, medium, highs. The levels may be assigned a weighted value to each level of complexity in order to make the measurement numerical. The measurement looks for a disproportionate amount of new mediums/highs versus in the requirements that were cancelled. If the new requirements have a much larger effort, this points to scope creep.
Level of effort of total number of requirements added since baselining	This can determine schedule and budgetary risks.	If there is a large level of effort being added, the schedule and budget may need to be adjusted to account for new work.	This is a common way projects lend themselves to overspending, especially on Time and Materials contracts. Instead of using the proper change control processes, oftentimes the business will change requirements that tack on additional scope.
Level of effort of total number of cancelled baselined requirements	This can determine potential savings in time (schedule) and budget due to work being cancelled.	If there is a large level of effort being removed, the schedule and budget may need to be adjusted to account for the removal of scope.	Projects can also use this metric in the Change Management process, as some changes may be able to repurpose hours reduced from the cancellation of requirements.

Metric	Use	Project Impact	Additional Notes
Number of hours spent on cancelled baselined requirements	This can be used to determine if there is too much time spent on requirements that are unattainable.	This can point to time (schedule) and budget that were used with no fruitful outcome.	This will point to the need to make requirements more granular so that they are attainable.
Number of test cases requirements are put through	This can be used to determine if there are too many test cases for a requirement.	This can point to requirements not being concrete enough, and may have multiple meanings.	This will point to the need to have a more appropriate level of detail and granularity so that they are more understandable.
Number of inquiries by developers on clarifying the requirements	This can determine if there is too much time being spent clarifying requirements.	If there are too many questions seeking clarification, a requirement may be too ambiguous for developers. There can also be scope creep or the requirement is not well written.	This can lead to a bigger issue of developers misinterpreting requirements, without asking for clarification. Should this occur, important functionality may be missed, and either quality will be impacted or additional rework will be required.
Number of requirements identified / added during the development	This can be used to determine if the requirements are not addressing every possibility	If there are too many requirements being identified or added, it can be an indicator that the requirements developed originally have not addressed everything required.	This can indicate that requirements were not written at a level that ensures completeness. This can lead to important functionality in the solution being missed, and either quality will be impacted or additional rework required.

Appendix A. Revision History

Table 9: Revision History

Revision	Description Of Change	Author	Date
1.0	Initial Draft for State Review	KPMG	March 11, 2016
1.1	Minor modifications including TennCare identity, new SDLC	KPMG	December 12, 2018
2.0	Include elaboration and traceability of requirements	KPMG	September 16th, 2019
2.1	Minor modifications based on feedback of elaboration and traceability of requirements.	KPMG	October 22nd, 2019
2.2	Minor modifications based on walkthrough of edits from feedback, as well as overall grammar cleanup	KPMG	November 7th, 2019
3.0	Updated this document for the IS RFP release	KPMG	June 12, 2020
4.0	Approved by TennCare for publishing to the bidder's library for procurement	KPMG	September 2020

Appendix B. Definitions, Acronyms and Abbreviations

Table 10: Glossary of Terms

Acronym	Definition
CMS	Centers for Medicare and Medicaid Services
COTS	Commercial Off-The-Shelf software
DDI	Design, Development, and Implementation
DED	Deliverable Expectation Document
EA	Enterprise Architecture
KPI	Key Performance Indicator
RTM	Requirements Traceability Matrix
SaaS	Software as a Service
SILC	TennCare Solution Implementation Lifecycle
SPMO	Strategic Project Management Office
SV	Solution Vendor
TARB	TennCare Architecture Review Board
TAS	Technical Advisory Services

Appendix C. Requirements Review Checklist

The following checklist can be used to validate that an individual project solution requirement is properly formed and can be validated. Use this checklist to validate the quality of each requirement created.

Table 11: Solution Requirement Review Checklist

Evaluation Criteria	Yes	No	Remarks
A test case is associated with the requirement.			
The requirement can be understood by affected parties (e.g., SME, developers, and testers).			
Unacceptable words and phrases are absent (e.g., adverbs, adjectives, as appropriate, at a minimum).			
Requirement conforms to standard format.			
Requirement is at the appropriate level of detail for its position in the hierarchy.			
Requirement has the associated information required by the Requirement Management Standard.			
Requirement is within scope.			
Requirement is terse.			
Requirement avoids specifying design.			
Requirement is feasible.			
Requirement is written in the imperative (shall).			
Cross-references are specific so the information can be easily located; the reference is located in the project document library if it is external to the requirement.			
Requirement can be traced to its parent or driver.			
Requirement is unrestrictive; it can be implemented by more than one solution or design.			
Assumptions and dependencies for requirements are stated.			
Requirement is written in the active voice.			
Requirement is individualized.			
Requirement is consistently granular.			
Requirement is written with consistent terms.			
Requirement does not combine two related, synonymous, or opposite terms.			
Requirement does not contain ambiguous boundary values.			
Requirement is written as a positive.			
Requirement is logically simple.			
Requirement is not missing exceptions.			

The following checklist can be used to ensure that the solution requirements on the aggregate are complete. Once all requirements have been developed, ensure that they, as a group, meet these criteria:

Table 12: Aggregate Solution Requirements Review Checklist

Evaluation Criteria	Yes	No	Remarks
Requirements are consistent with each other.			
Requirements are complete: every case or scenario is addressed.			
Requirements address user interfaces.			
Non-functional requirements are addressed.			
Requirements address system and user error conditions.			
All requirements are traced to their parent or driver (no dropped traceability).			
Interfaces are specified (internal/external).			

Appendix D. Requirement Traceability Semantic Model

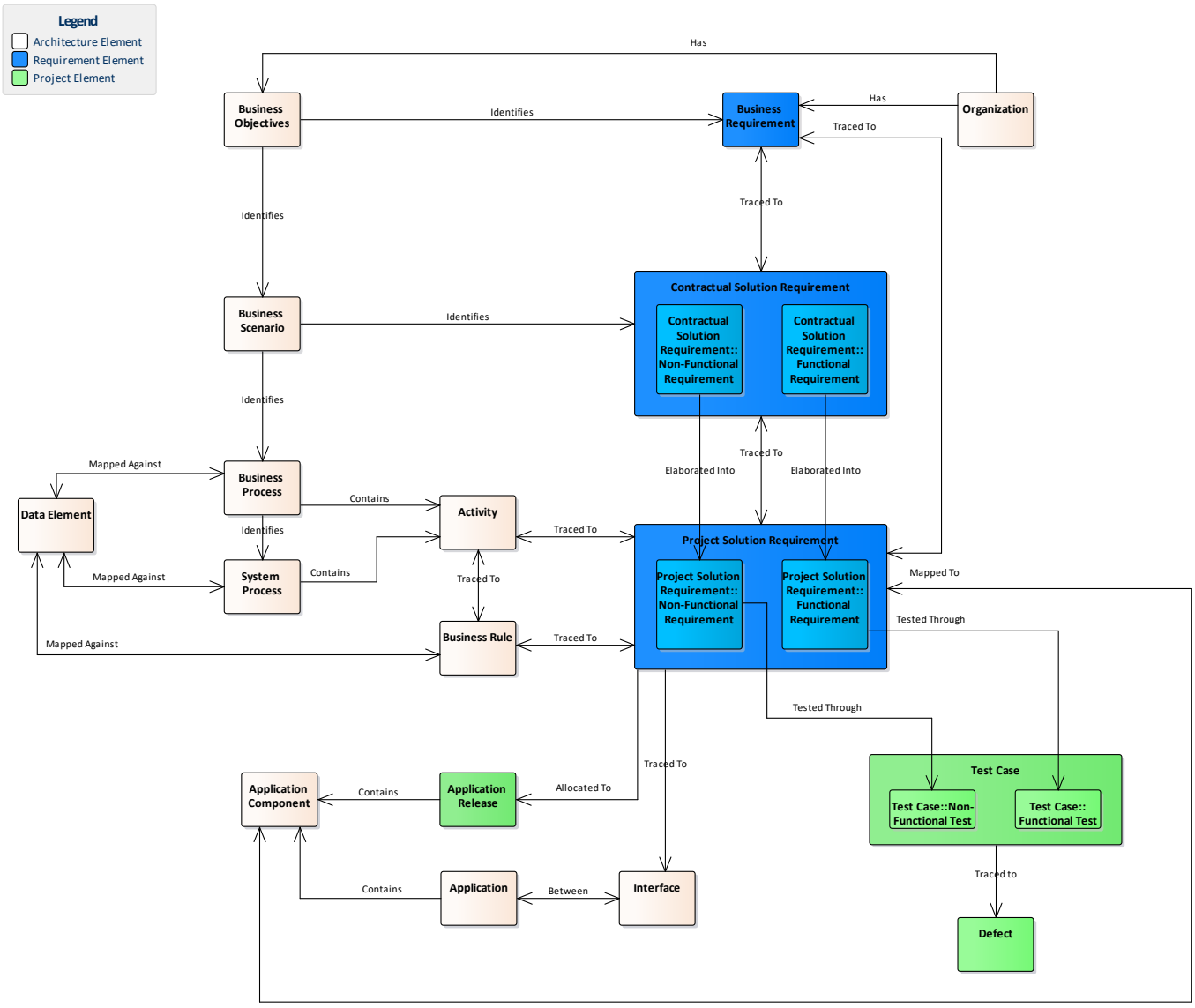

Figure 9: Requirement Traceability Semantic Model

Figure 9: Requirement Traceability Semantic Model illustrates a logical model for how requirements traceability can be implemented into a requirements and architecture repository. This model may not apply to each individual TennCare systems project.

This traceability is required to prove that an implemented system aligns with achieving an organization's goals and objectives.

The model in *Figure 9: Requirement Traceability Semantic Model* also illustrates that an organization has business objectives (which are the goals and outcomes that a business is trying to achieve). Business objectives uncover business requirements, and drive writing business scenarios which identify contractual solution requirements (functional and non-functional). Business scenarios (or user stories) identify business process models, which contain activities and decompose into system process models.

Business rules, data elements, and activities are all mapped to each other and to project solution requirements (functional and non-functional). Project solution requirements are allocated to application releases.

Application releases contain application components, which are mapped to solution requirements. Interfaces between applications will also have project solution requirement.

Project solution requirements are related to contractual solution requirements and business requirements in a requirements hierarchy. Functional and non-functional contractual solution requirements are elaborated into project solution requirements.

The project solution requirements are traced against the functional and non-functional test cases, and through those, to defects.

Therefore, a path from business objectives to an implemented solution can be as follows:

Business Objectives -> Business Requirements -> Contractual Solution Requirements -> Project Solution Requirements -> Test Application Component -> Implement Application Component