

# Your Document is Always Greener on the Other Side

## Applying Analysis Rigor Using Compliance Gateways

John Cheesman, Strata Software

### Table of Contents:

1. Lost in Translation.....	2
2. Rigor with Readability.....	3
3. Structure in Practice.....	4
4. Through the Gateway.....	6
5. Scaling Up.....	8
6. Summary.....	9

# 1 Lost in Translation

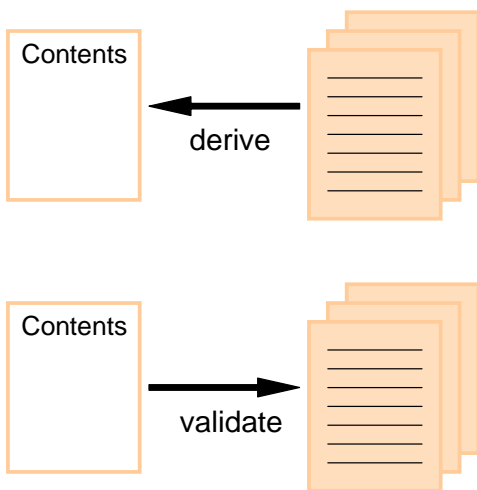
**Business analysis** and software specification are closely related disciplines within the software development lifecycle but they have a communication problem. Business analysis is predominantly a business-facing activity where the priority is on communication between IT and the business. Consequently, deliverables in this area are highly document-centric and often lack rigor.

Software specification on the other hand has the primary objectives of precision and detail. This detail is often captured in models and tools which provide rigor, but the resulting models can be difficult to communicate. Much information can be carefully crafted into the model never to resurface.

It remains a common metric of software project assessments that many of the errors occurring with software development projects arise on the boundary between business requirements analysis and software specification. This boundary is where the language changes with the effect that business concepts, objectives and priorities get lost in translation.



# 2 Rigor with Readability



**Analysis artifacts** such as use cases, business rules and information models are inter-related and naturally form a network of relationships. These relationships can readily be captured using modeling tools, but due to their complexity the models created can be difficult to read, understand, review or validate. Many tools therefore provide reporting facilities to allow you to extract the information you captured back into document form. By adopting a hierarchical form, documents provide a simple, structured basis for organizing artifacts and understanding their relationships. If we can provide improved structural support to documents we can combine the benefits of modeling with the readability of documents and reduce the loss of information on the business-IT boundary.

Normally in documents the “Contents” are derived from the content. Indeed, Microsoft Word™ provides a specific tool for generating a Table of Contents from the contents. This provides consistency between the two, but does not provide a quality or completeness check on the content. Similarly, document templates can ensure a document begins life in the right way, but can’t easily be used to check it stays on track – people often delete those sections they’re not too clear on or have no information about, reducing the scope of their document. Switching the dependency round, from the Contents being derived from the content, to the Contents being used to validate the contents, provides an opportunity to use the Contents structure as a compliance and completeness check on the document contents.

VisibleThread enables us to do this through its concept of best practice compliance, and this opens the door for the document to gain rigor and become a form of model.

# 3

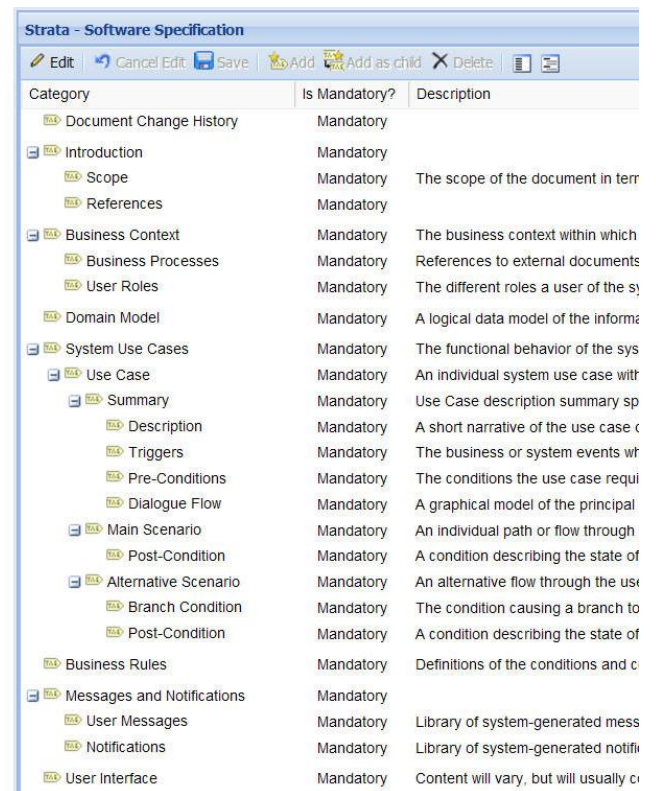
## Structure in Practice

At **Strata Software** we have a standards library for analysis artefacts and documents. Here we'll look at the software specification document which is centered on use cases and their details. Use Cases have a number of important properties, and we would like to ensure that they are captured and not overlooked. For example, each use case must have a summary description, a trigger and a pre-condition, and then a main success scenario and one or more alternative scenarios. Each scenario must have a post-condition, and each alternative scenario must define the condition causing that branch to be followed. Additionally, use case details will reference messages, business rules and notifications that we wish to capture independently of the use case body. Further, the user interface description may be included.

In VisibleThread we can represent this standard as a structural Best Practice and then use that best practice to check the compliance of our software specifications. An example Best Practice for Software Specification is shown to the right. The structure, properties and relationships of a software specification, described informally above, have now been encoded into a standard form allowing programmatic application.

For example, the Best Practice shown defines that a software specification must have a section for System Use Cases, within that separate sections for individual Use Cases, and within each use case section a Summary section and then different Scenario details.

We can now apply such a structure to individual documents to validate their compliance. Let's follow an example use case document under construction and see the effect.

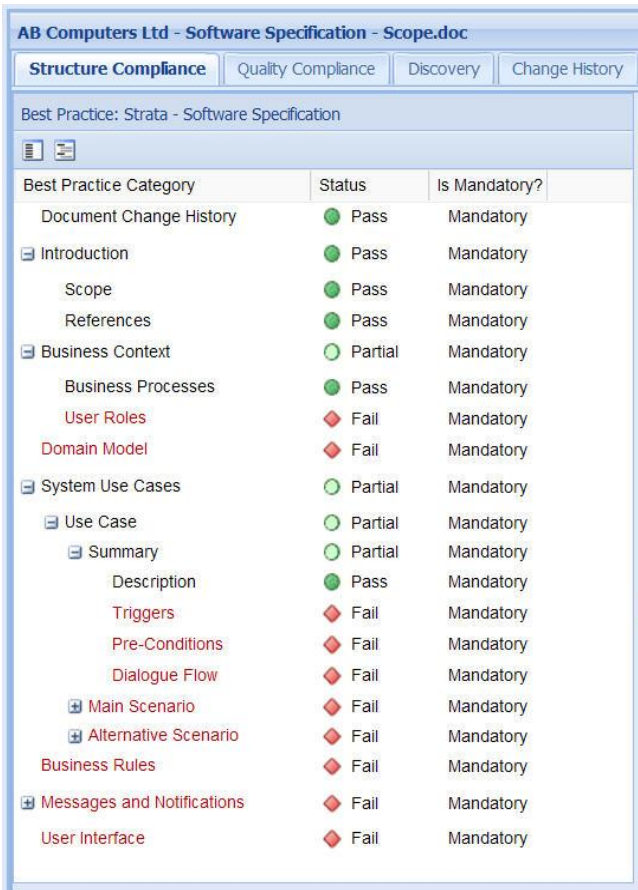


The screenshot shows a software specification tool interface titled "Strata - Software Specification". It features a menu bar with options: Edit, Cancel Edit, Save, Add, Add as child, and Delete. Below the menu is a table with three columns: Category, Is Mandatory?, and Description. The table lists various categories and their mandatory status, along with brief descriptions.

Category	Is Mandatory?	Description
Document Change History	Mandatory	
Introduction	Mandatory	
Scope	Mandatory	The scope of the document in terr
References	Mandatory	
Business Context	Mandatory	The business context within which
Business Processes	Mandatory	References to external documents
User Roles	Mandatory	The different roles a user of the sy
Domain Model	Mandatory	A logical data model of the inform
System Use Cases	Mandatory	The functional behavior of the sys
Use Case	Mandatory	An individual system use case with
Summary	Mandatory	Use Case description summary sp
Description	Mandatory	A short narrative of the use case c
Triggers	Mandatory	The business or system events wr
Pre-Conditions	Mandatory	The conditions the use case requi
Dialogue Flow	Mandatory	A graphical model of the principal
Main Scenario	Mandatory	An individual path or flow through
Post-Condition	Mandatory	A condition describing the state of
Alternative Scenario	Mandatory	An alternative flow through the use
Branch Condition	Mandatory	The condition causing a branch to
Post-Condition	Mandatory	A condition describing the state of
Business Rules	Mandatory	Definitions of the conditions and c
Messages and Notifications	Mandatory	
User Messages	Mandatory	Library of system-generated mess
Notifications	Mandatory	Library of system-generated notifi
User Interface	Mandatory	Content will vary, but will usually c

# 3

## Structure in Practice



The screenshot shows a software specification document titled "AB Computers Ltd - Software Specification - Scope.doc". The document is organized into a table with three columns: "Best Practice Category", "Status", and "Is Mandatory?". The table lists various sections of the document and their compliance status. The status is indicated by a colored circle: green for "Pass", yellow for "Partial", and red for "Fail". The "Is Mandatory?" column indicates whether the section is required.

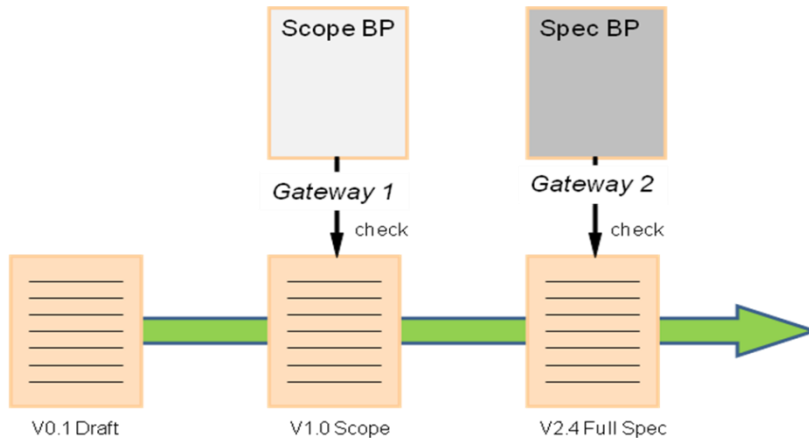
Best Practice Category	Status	Is Mandatory?
Document Change History	Pass	Mandatory
Introduction	Pass	Mandatory
Scope	Pass	Mandatory
References	Pass	Mandatory
Business Context	Partial	Mandatory
Business Processes	Pass	Mandatory
User Roles	Fail	Mandatory
Domain Model	Fail	Mandatory
System Use Cases	Partial	Mandatory
Use Case	Partial	Mandatory
Summary	Partial	Mandatory
Description	Pass	Mandatory
Triggers	Fail	Mandatory
Pre-Conditions	Fail	Mandatory
Dialogue Flow	Fail	Mandatory
Main Scenario	Fail	Mandatory
Alternative Scenario	Fail	Mandatory
Business Rules	Fail	Mandatory
Messages and Notifications	Fail	Mandatory
User Interface	Fail	Mandatory

**In the example**, we're specifying a software system which allows for the placing, tracking and fulfillment of orders for new computers from AB Computers Ltd. A specification has been put together and distributed for review. It contains a host of use cases, some good descriptions, and the business process context. It's a good overview or scoping document. But what's missing?

If we apply the Software Specification Best Practice to the document it measures its standards compliance and highlights which areas Pass, Partially Pass or Fail their compliance check. This is a valuable indicator of omissions, errors and inconsistencies, but also, importantly, a guide to what to do next. For example, we can see that there is no Domain Model and there are no formalized Business Rules so these will need to be put in place.

From the rolled-up dashboard summary we can also gauge an overall measure of document compliance which gives us an indication of how much work is left to do.

# 4 Through the Gateway



**Compliance checks** naturally dovetail with governance processes. We can tune the structural best practice checks we have demonstrated to align with a required document quality and completeness gateways as part of a governance process. These gateways define what level of compliance is needed at what point in the process. For example, our software specification document seemed in good shape from an initial scoping point of view, but is short of lots of detail. At one point in the process it might be compliant, but at a subsequent point the same document would fail compliance as we move the bar up.

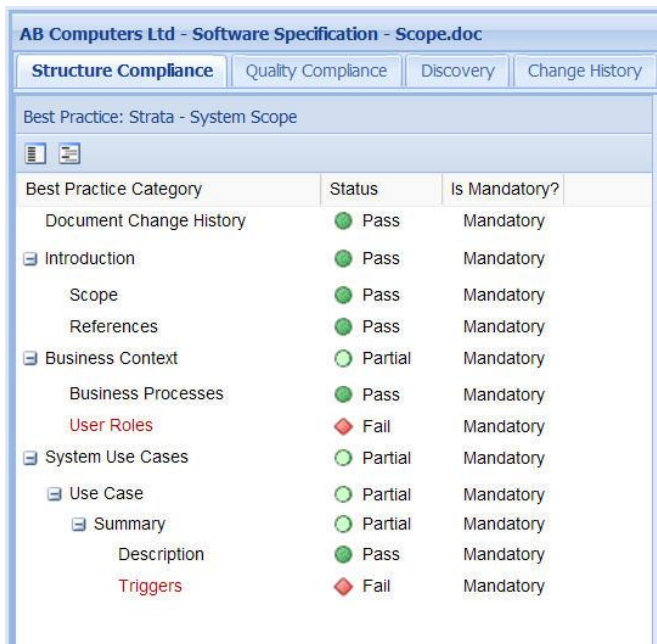
We can address this by defining different Best Practices, each representing a different stage in the life of the document. In our example we define a System Scope Best Practice and a related, more detailed Software Specification Best Practice.

For a scoping document we're interested in the big picture and the business context, but we're pretty relaxed about the details. It's early in the project and we're just trying to get an idea of the functional

scope of the system. We want the header information defined, but most of the detail is unnecessary, even detrimental as it represents potentially wasted effort at this stage.

If we now associate the scoping document with a reduced Scope Best Practice we get much higher degrees of compliance. The Scope Best Practice only mandates the definition of the Business Context and the identification of system use cases and their triggers.

# 4 Through the Gateway



AB Computers Ltd - Software Specification - Scope.doc

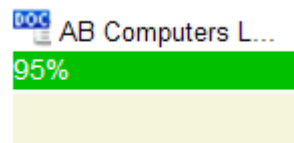
Structure Compliance | Quality Compliance | Discovery | Change History

Best Practice: Strata - System Scope

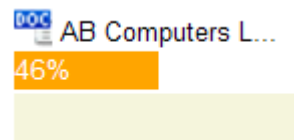
Best Practice Category	Status	Is Mandatory?
Document Change History	Pass	Mandatory
Introduction	Pass	Mandatory
Scope	Pass	Mandatory
References	Pass	Mandatory
Business Context	Partial	Mandatory
Business Processes	Pass	Mandatory
User Roles	Fail	Mandatory
System Use Cases	Partial	Mandatory
Use Case	Partial	Mandatory
Summary	Partial	Mandatory
Description	Pass	Mandatory
Triggers	Fail	Mandatory

In our example, the user roles have been overlooked within the business context, which could mean we would miss some important stakeholders, so this needs to be fixed. Also, we have identified use cases but failed to call out the triggers or events which give rise to those use cases so we need to identify and include those. This is a useful check point to help validate the quality of the use case set at the scoping stage.

With this best practice, compliance now comes in at 95%, so things are looking greener.



As we pass through the gateway we now switch to the more thorough software specification best practice and initially compliance will drop to 49% as before, with the failure points being sign posts towards which areas need to be specified. We then update the document in those directions and move to the next gateway.



In this way we can adapt compliance gateways to include simple automated checks on documents which validate their contents against some pre-defined content standards.

# 5 Scaling Up

**Looking more broadly**, larger projects often have many different documents, each tackling some subset of the project in parallel. Applying a common set of best practice to a family of documents yields not only individual document quality improvement, but quality improvements to the document set as a whole.

Our AB Computers project demonstrates this higher level view with the following dashboard metrics for a complete set of software specifications for a given project. In this example we are validating the whole specification set against the System Scope best practice. We can see the average compliance (64%) as well as the individual document variations.

Compliance gateways can be applied as before, but when operating at the document set level can become project gateways monitoring the passage of a project from one phase to another. For example, a governance process may mandate that all specification documents must meet a certain compliance threshold before going to tender with suppliers.

Doc Set / Document / Heading	Best Practice	Size	Structure Compliance
AB Computers - Project ABC	Strata - System Scope	3 documents	64%
Software Specification A.doc	Strata - System Scope	177 paragraphs	56%
Software Specification B.doc	Strata - System Scope	167 paragraphs	95%
Software Specification C.doc	Strata - System Scope	134 paragraphs	41%

# 6 Summary

**In this paper** we have addressed a number of topics and shown how the concept of structured best practice compliance can be utilized to improve document quality and project governance. At a single document level, structural compliance allows modeling rigor to be injected into the analysis while maintaining the communication benefits of a document. We then elaborated on this idea to introduce the notion of different compliance points for a document during its lifecycle, providing structured guidance for document improvement and evolution. Finally we scaled up to the project level to demonstrate the usage of document *set* compliance and its applicability to project governance and phase transition management.

## About Strata Software

Strata Software is a leading provider of Requirements Definition and Management services and solutions. With an average of 20 years experience in the field, our consultants specialise in requirements-driven improvement and optimisation of application lifecycle processes. We establish customer-oriented best-practice in requirements definition and management, tool usage and project lifecycle integration, providing measurable business value in aligning software solutions to business requirements.

## About VisibleThread

VisibleThread helps corporate IT departments create superior project documentation that leads to successful projects. Our document structure and quality analysis tools, combined with the ability to create tailor-made best practices documents, provide customers with the insight and metrics they need to drive IT lifecycle process improvement and make better decisions throughout the project lifecycle. VisibleThread ensures a uniform approach to IT documentation resulting in consistency across documents, higher quality outputs and lowered cost.

### Strata Software

Strata Software Limited  
Ashcombe Court, Woolsack Way,  
Godalming, Surrey GU7 1LQ,  
United Kingdom

+44 (0)1483 422515  
[info@stratasoftware.com](mailto:info@stratasoftware.com)  
[www.stratasoftware.com](http://www.stratasoftware.com)

### VisibleThread

1101 E. 33rd Street  
3rd Floor, Suite #C300  
Baltimore, MD 21218

(443) 451-7005  
[info@visiblethread.com](mailto:info@visiblethread.com)  
[www.visiblethread.com](http://www.visiblethread.com)