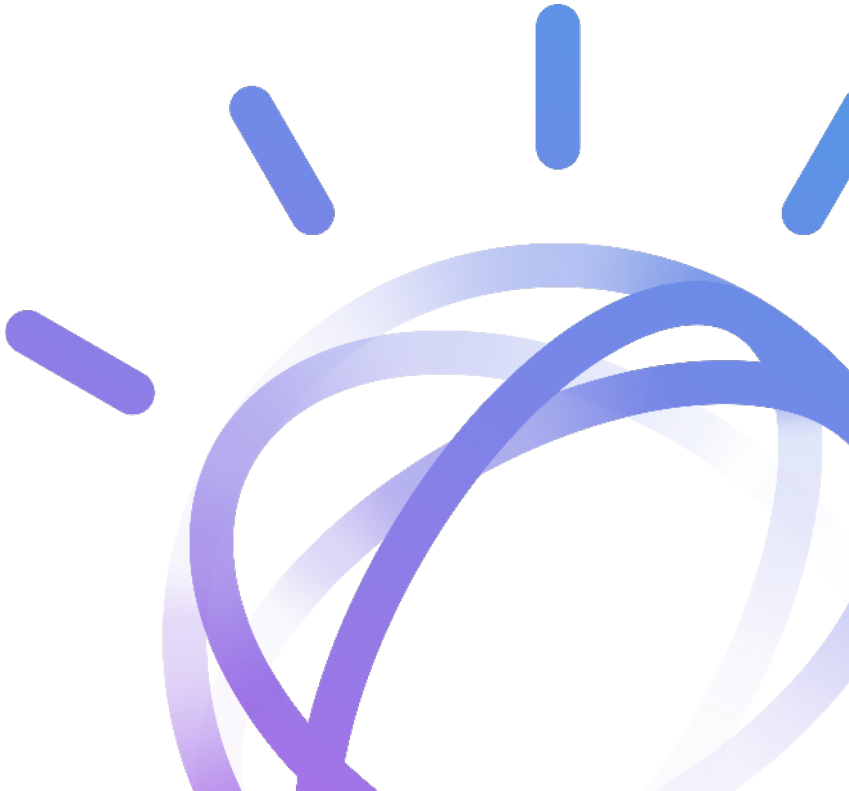
IBM AI for Systems Engineering

Watson IoT

Enrico Mancin
Lead Architect Europe
Watson IoT

IBM



About Today's Presenter

Enrico Mancin
(enrico.mancin@it.ibm.com)

Bio:

Enrico Mancin is the CE Tech Lead and IoT Lead Architect Europe IBM Watson IoT and the President of AISE INCOSE Italy Chapter.

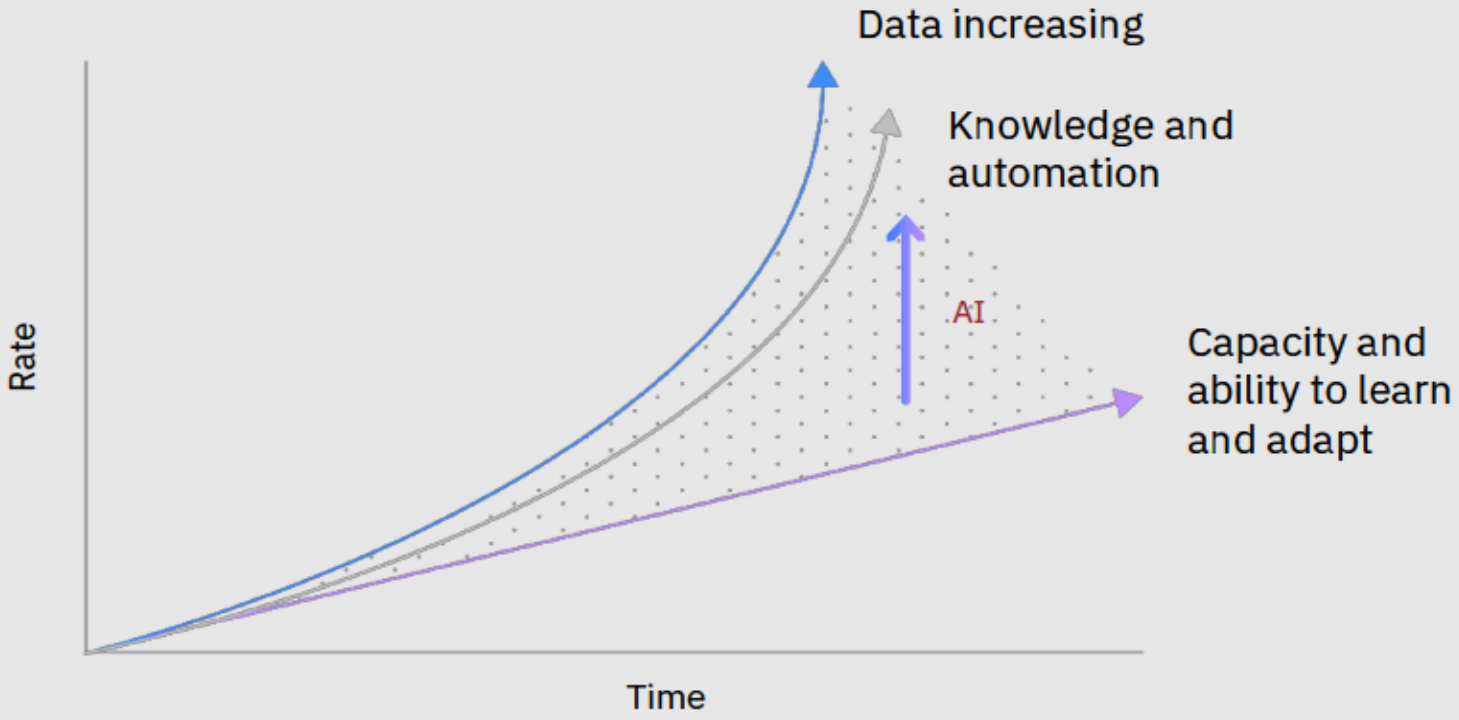
He is a former Business Solution Professional engineer first in the Industrial and then in the Public Sector of IBM Italy, where he was the lead systems engineer for some of IBM's development projects. On behalf of IBM, he has led client engagements in aerospace and defense, system development and IT enterprise architecture, helping clients transform their engineering organizations using IBM technologies, methods and tools.

He has been a practitioner, consultant, author and speaker on systems engineering and software development methods for 30 years. While an engineer, project manager, chief architect in important Italian companies, his experience spans in project management, systems engineering, architectural modeling and requirements analysis. His current specialization includes model-driven system development, enterprise architecture, estimation methods and solution architecture.

Agenda

- AI helps companies increase the capacity of their employees through automation and knowledge
- Engineering knowledge has become big data
- IBM’s vision to inject automation and intelligence throughout the engineering lifecycle
- What we are doing today
  - Pre-training Watson
  - Released our first AI offering in requirements quality

AI accelerates our capacity to act by injecting knowledge and automation into business processes



One AI Example: Increasing Capacity through Knowledge at Woodside

Example of leveraging AI:
Working with Watson, Woodside Energy built a customized tool that allows its employees to find detailed answers to highly specific questions—even on remote oil and gas facilities.

- 38,000 Woodside documents were used to train the solution—this would take a human over five years to read
- 30 years of practical engineer experience at the fingertips of all Woodside employees
- 75% decrease in time employees spend searching for expert knowledge

We teach Watson to think like an engineer.

Watson teaches us to think like a thousand engineers.

woodside

One AI Example: Increasing Capacity through Automation at Autoglass

Example of leveraging AI:

Autoglass built the world's first automated vehicle body damage assessment and quote generation system. Using the IBM Watson Visual Recognition service, the system analyzes the photos that customers upload when they submit a body damage claim, applying the same classification logic as Belron's highly experienced damage assessment advisors.

- Shortens the claims cycle for certain damage assessments by more than 95%
- Enriches client experience through personalized conversation flows and real-time answers
- Enables damage assessment advisors to focus on more complex damage claims

AUTOGLASS®

- Shortens the claims cycle for certain damage assessments by more than 95%
- Enriches client experience through personalized conversation flows and real-time answers
- Enables damage assessment advisors to focus on more complex damage claims

"We're able to automate nearly half of our body damage claims assessments using image recognition technology, and we expect that rate to get even higher."

Dafydd Hughes
IT Manager at Autoglass Body Repair

Engineering knowledge has become big data... driven by the complexity of product development

Automotive

150M Lines of code in new Ford F-150 Truck from 155K in 2003

100x increase in test cases

40% of total IT Budget spent on QA and testing by 2019

Electronics

12M lines of code in mobile phone

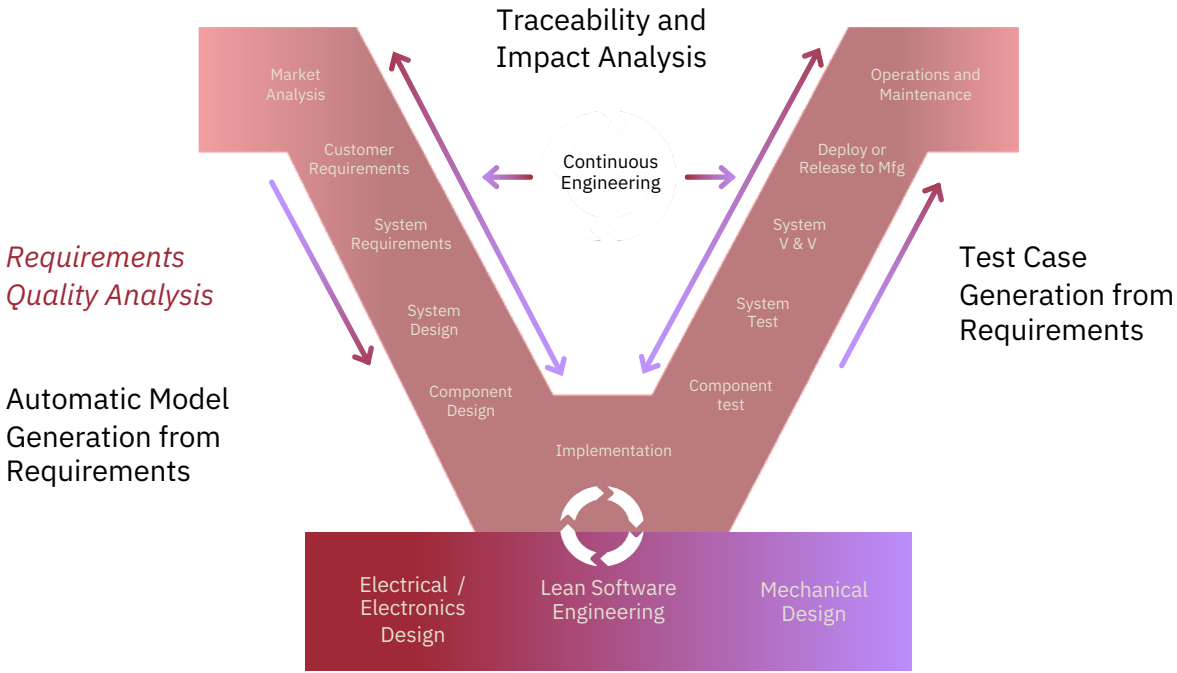
1.4M lines of code in robotic surgical system

Aerospace and Defense

90% of F-35 fighter jet functionality is software driven

Optimize engineering with AI

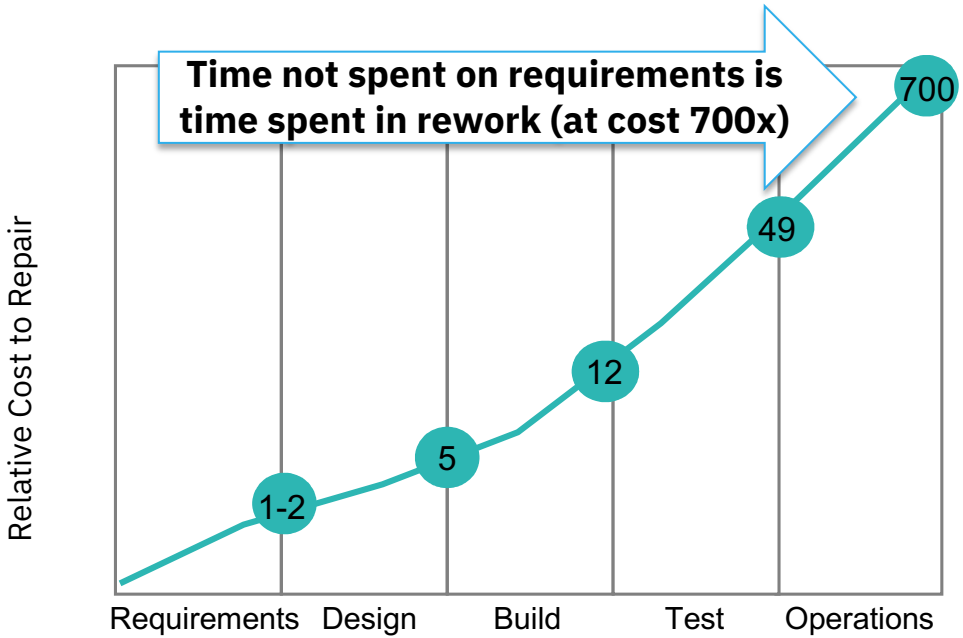
Inject automation and intelligence across the engineering lifecycle



Companies with poor requirements management 3 project failures for every 1 success

Companies with poor requirements management, on average, spend $2.24 million more per project

The cost to fix software defects rises exponentially with each successive phase of the project life cycle



Current solutions…and why they can miss errors

- **Rules engines**
  - Simple key word searches cannot understand words in context
    - Ex. of “clear”
    - Ambiguous: “The GPS system shall provide a clear perspective of the road”
    - Not ambiguous: “The GPS System shall clear the display on transition to power off mode”
- **Peer review**
  - Difficult to enforce across teams
  - Manual, tedious review of long documents
- **Checklists**
  - Limited capacity to keep track of all quality indicators (cognitive span is 7 +/- 2 things)

Best Practice Guidelines

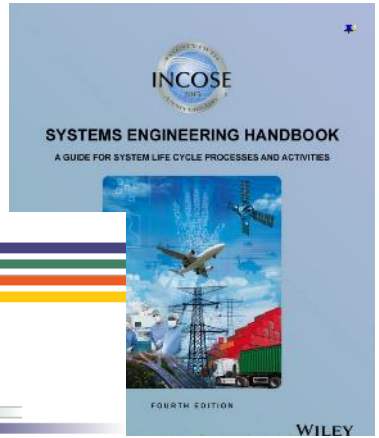
INCOSE Guide for Writing Requirements

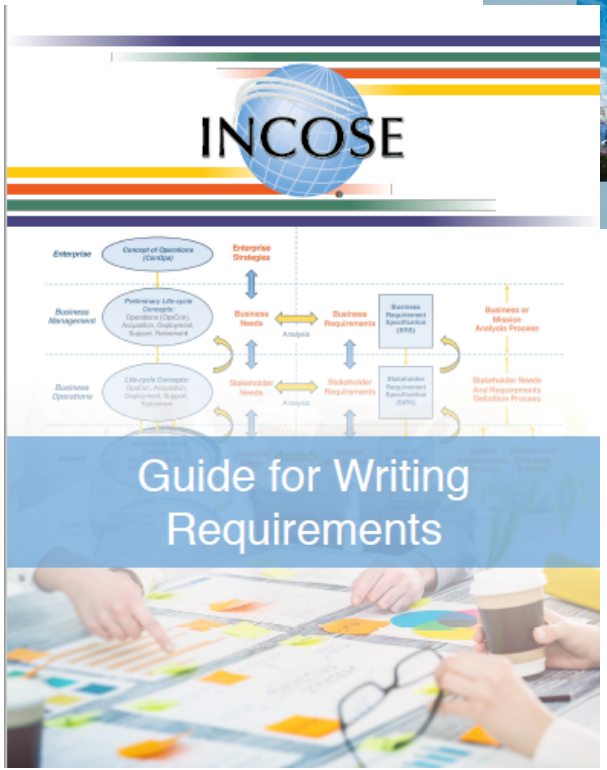
Written by a worldwide cross-industry team

Rules reflected in the Systems Engineering Handbook and ISO 15288

Some rules submit to a dictionary check

Some are more subtle





Types of Requirements

- **User Requirements** - define the results the users expect from the system

"The homeowner shall hear an alarm when smoke is detected"

- **System Requirements** - define what the system must do to satisfy the users

"The alarm shall produce a sound between 125-155 dBA"

- **Design Requirements** - define all of the components necessary to achieve the system requirements

"The alarm shall be produced by part # 123-45-678"

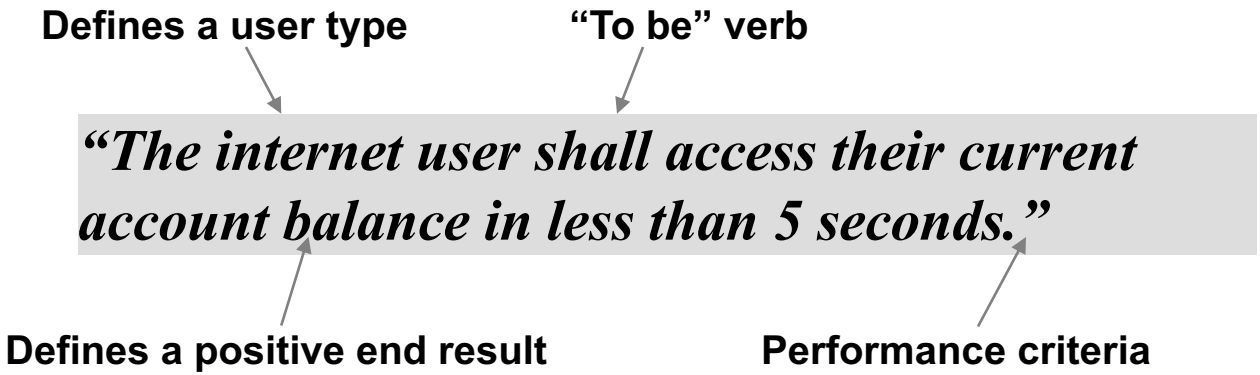
Writing a requirement

- Complete sentence
- States subject and predicate
  - Subject is a user type or the system under discussion
  - Predicate is a condition, action, or intended result.
- Consistent use of language
- Specifies:
  - desired goal or result (User requirement)
  - function (System requirement)
  - constraint (either)
- Contains a success criterion or other measurable indication of the quality.

Language

- Use consistent language, for example:
  - “Shall,” “will” or “must” are mandatory
  - “Should” is optional, but omission must be justified
  - “May” is desirable
- Use consistent terminology
  - Define terms – use a Glossary
  - Avoid using the same name for different things
  - Avoid using different names for the same thing

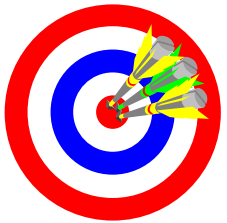
Anatomy of a Good User Requirement



- This requirement sentence identifies a specific user and end result that is wanted within a specified time.
- It also defines the success criteria in measurable terms - "access ... account balance" "in less than 5 seconds."

The challenge is to seek out the user type, end result, and success measure in every requirement you define.

Characteristics of a Good Requirement



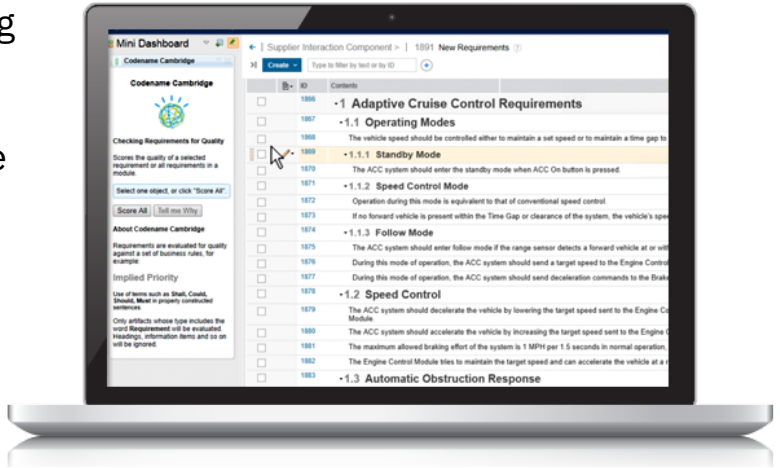
Each individual requirement should be:

- Correct - Technically and legally possible
- Complete - Express a whole idea or statement
- Clear - Unambiguous and not confusing
- Consistent - Not in conflict with other requirements
- Verifiable - It can be determined that the system meets the requirement
- Traceable - Uniquely identified and can be tracked
- Feasible - Can be accomplished within cost and schedule
- Modular - Can be changed without excessive impact
- Design-Free - Does not pose a specific solution on design (i.e., implementation free)
- Positive - Written in the affirmative, not the negative

IBM Requirements Quality Assistant

New Watson capability embedded inside DOORS Next Generation (DNG)

- Removes risk and ambiguity in the requirements authoring phase out-of-the-box by using AI (Watson Natural Language Understanding)
- Pre-trained to detect key quality indicators designed to be consistent with the **INCOSE Guidelines for Writing Good Requirements**
- Authors receive coaching from Watson to improve the quality of the requirement as it is being written



Enterprise benefits (400 engineers example)

- Reduce the cost of defects by 60% to save $3.9M
- Reduce cost of manual reviews by 25%
- Retain engineering expertise for junior engineers

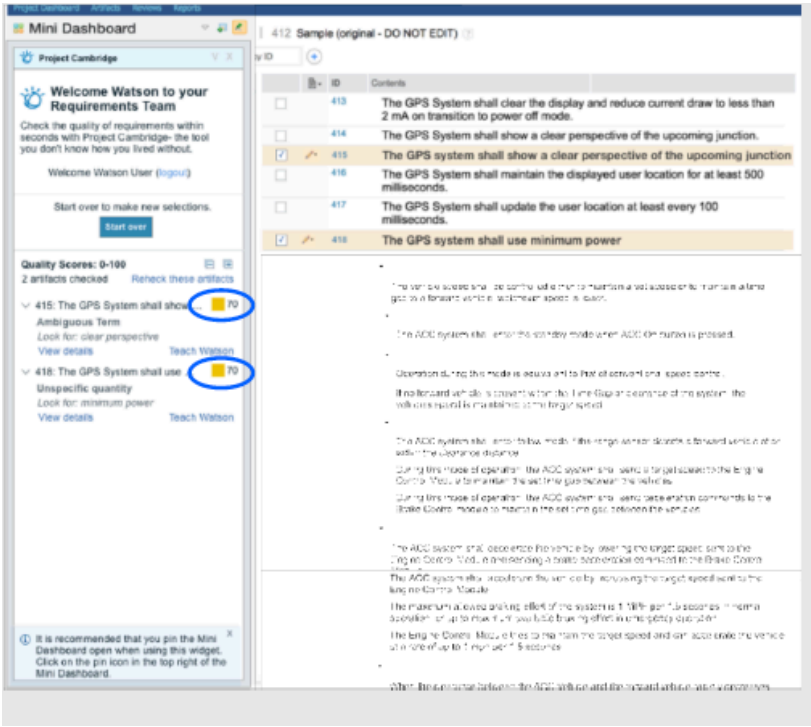
IBM Requirements Quality Assistant

- Grades requirements against a criteria that was designed to be consistent with the INCOSE Guidelines for Writing Good Requirements

- Pre-trained to detect 10 quality issues
  - Unclear actor or user
  - Compound requirement
  - Negative requirements
  - Escape clause
  - Missing units
  - Missing tolerances
  - Ambiguity
  - Passive
  - Incomplete requirements
  - Unspecific quantities

Table 1. Deductions for each issue

Issue	Deduction
Compound requirement	30
Escape clause	10
Incomplete: Missing action	30
Incomplete: Missing actor	30
Incomplete: Missing object	30
Missing imperative	30
Missing limits	20
Missing unit	20

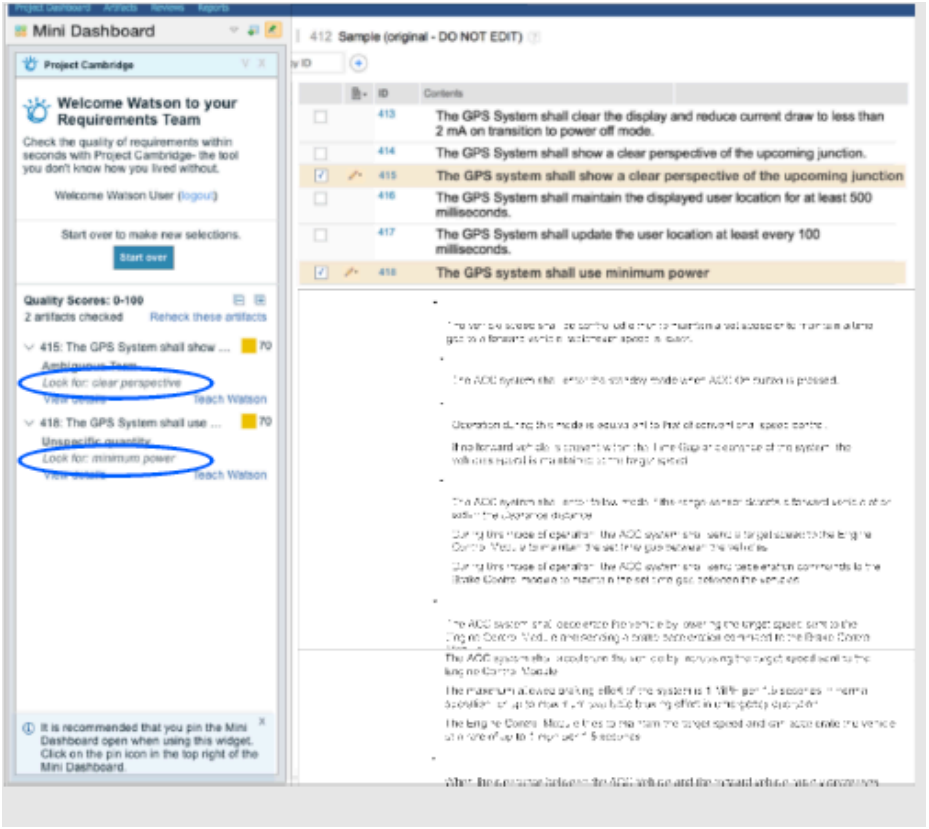


• Add to the list of quality issues or do deeper training through a 3 week services engagement with IBM services team

RQA Knowledge Center

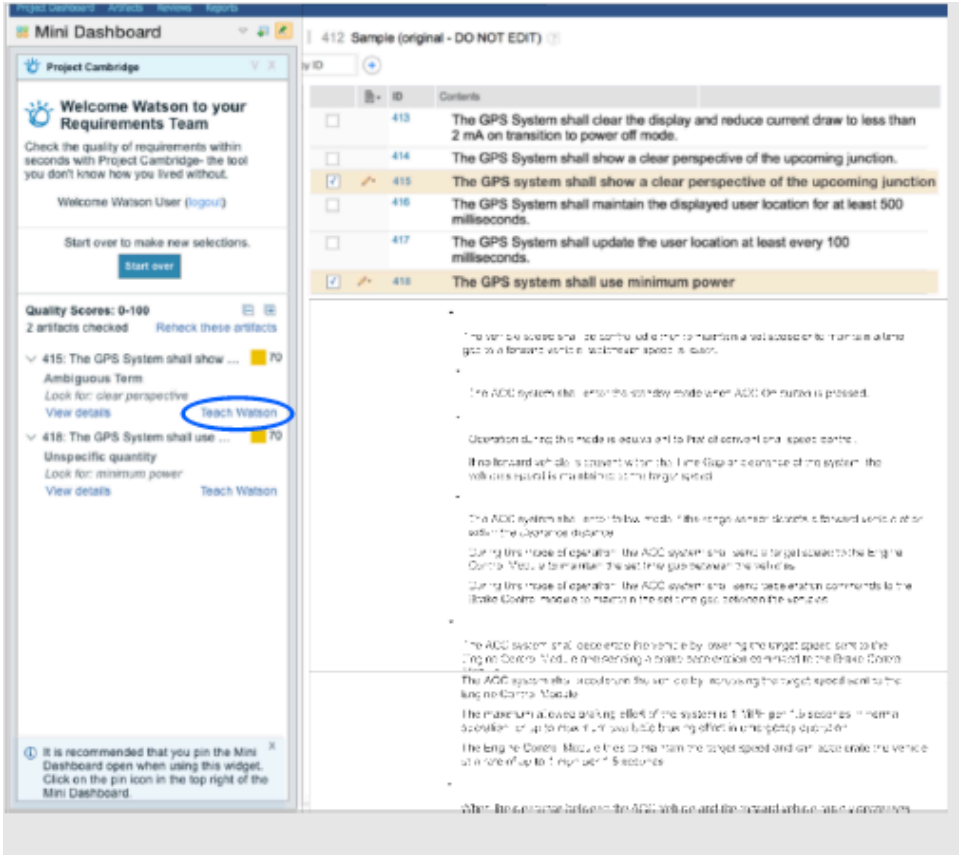
IBM Requirements Quality Assistant

- Identifies exactly what's wrong with the requirement
- Displays the issue to the requirements engineer



IBM Requirements Quality Assistant

- Learns from the requirements engineer
- Becomes “smarter” over time



Thank you