Digital Engineering

# Anforderungen im Wandel – oder kommen die Anforderungen in Zukunft von der KI und aus Modellen?

Peter Schedl
Program Manager
IBM Engineering Lifecycle Management
peter.schedl@de.ibm.com



watsonx™

# Questions raised during this Conference

Is MBSE (or UML) dead?

Warum modellbasiert arbeiten?

Brauchen wir noch Requirements Engineers (oder reicht KI) ?

Wie kann Generative KI das Requirements Engineering sinnvoll unterstützen?

KI als Co-Pilot, Chief Engineer oder Chaos?

Was sind die Risiken von KI?

ChatGPT, ChatGPT, ChatGPT...?

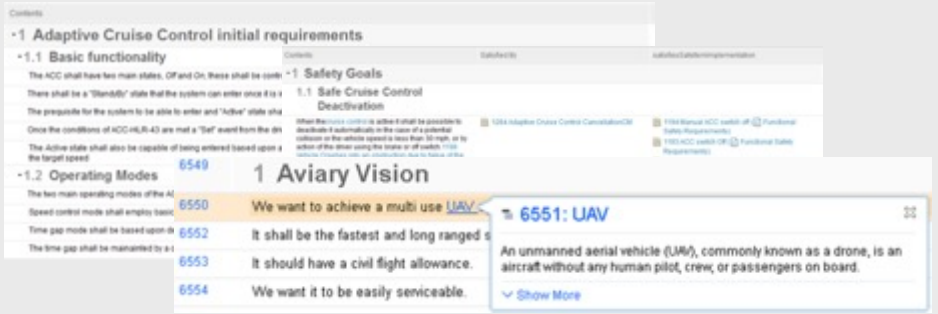
# Questions to be answered

- What is Requirements Engineering ?
  - Does AI help ?
  - Can AI generate Requirements ?
  - Does Modeling help ?
- What is MBSE and how does it relate to Requirements Engineering ?
  - What is a MBSE best-practice?
  - Can AI generate Models ?
- Do Models generate Requirements ?

# Requirements Definition and Management

Requirements Engineering involves:

- Requirements elicitation – gathering requirements from stakeholders/customers
- Requirements analysis and negotiation – check clarity, completeness, resolve issues
- Requirements specification – document with text, maybe with use cases and scenarios
- Functional analysis
- Requirements validation
- Requirements management – continuous activity, traceability, change handling



# Requirements Challenges

Misunderstood requirements by stakeholders and engineers

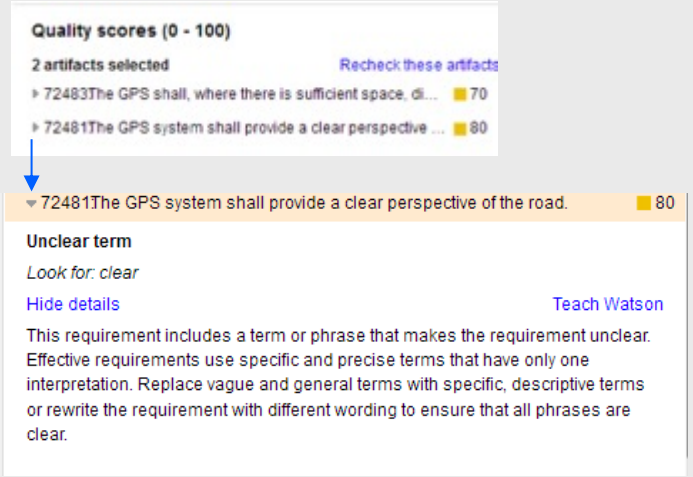
Poorly expressed requirements → RQA: AI based quality checker

Misunderstanding or omission by development

Missed test coverage

Requirement change impact misunderstandings

Little reuse



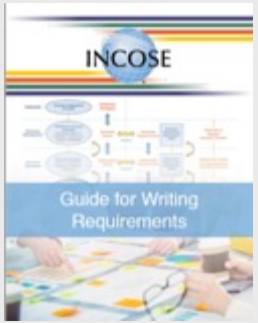
# IBM Engineering Requirements Quality Assistant

Watsonx capability embedded inside DOORS & DOORS Next introduced early **2019**

Removes risk and ambiguity in the requirements authoring phase out-of-the-box by using AI

Authors receive coaching from **Watson** to improve the quality of the requirement as it is being written

Pre-trained with the INCOSE Guidelines for Writing Good Requirements




Examples:

- Unclear actor or user
- Compound requirement
- Negative requirements
- Escape clause
- Missing units or tolerances
- Ambiguity
- Passive
- Incomplete requirements
- Unspecific quantities

# IBM watsonx

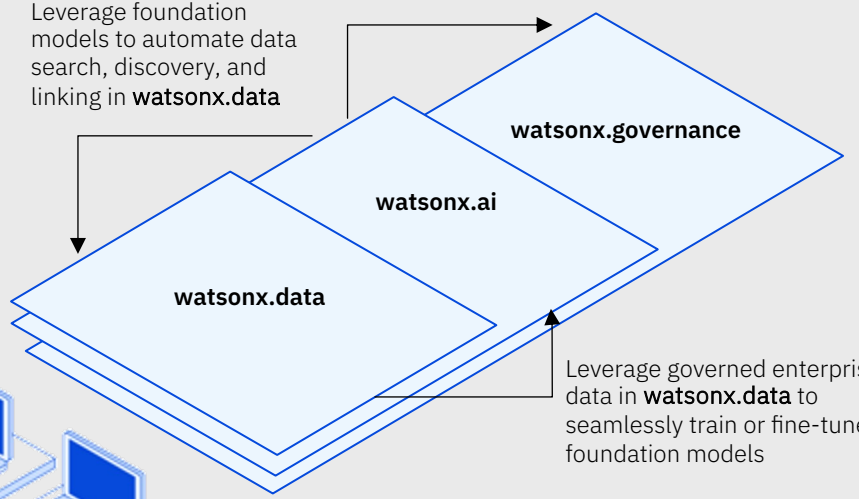
## The platform for AI and data

Scale and accelerate the impact of AI with trusted data.

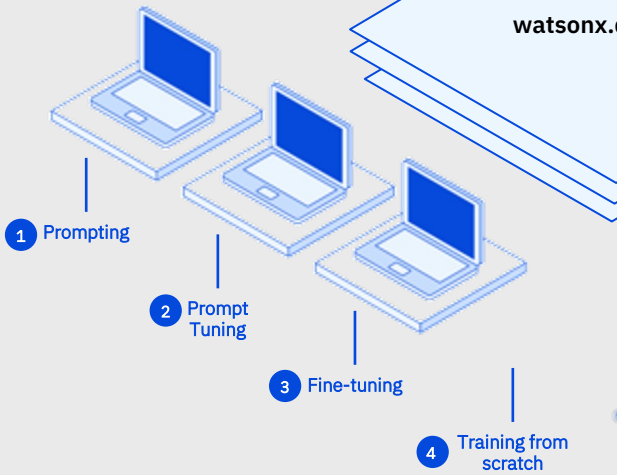
**watsonx**
Enterprise-ready generative AI and data platform

Enable fine-tuned models to be managed through market leading governance and lifecycle management capabilities

Leverage foundation models to automate data search, discovery, and linking in **watsonx.data**




Leverage governed enterprise data in **watsonx.data** to seamlessly train or fine-tune foundation models




**watsonx.data**

Scale workloads, for all your data, anywhere

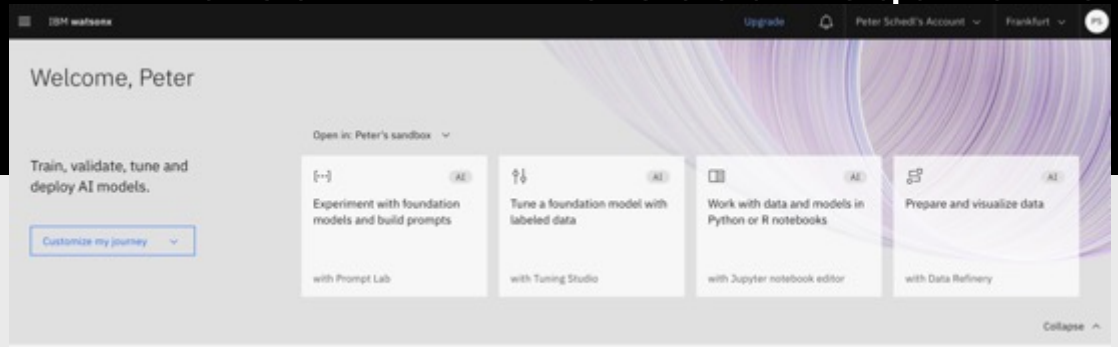
**watsonx.ai**

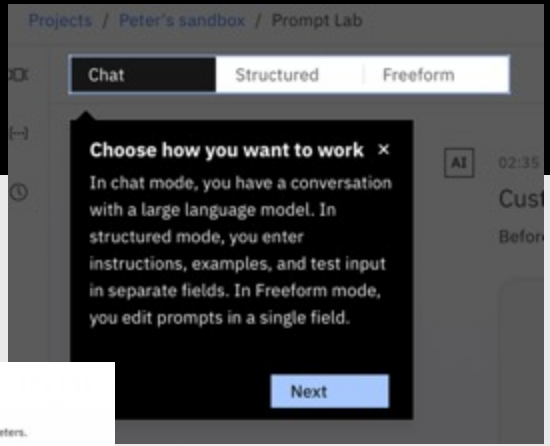
Train, validate, tune and deploy AI models

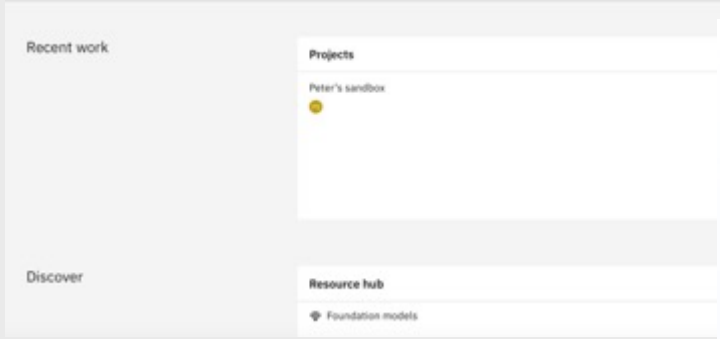
**watsonx.governance**

Enable responsible, transparent and explainable AI workflows

# IBM watsonx – Write Good Requirements





https://www.ibm.com/watsonx



Using the granite-13b-chat-v2 LLM

IBM w

PS You: 02:43 PM

How should requirements be written?

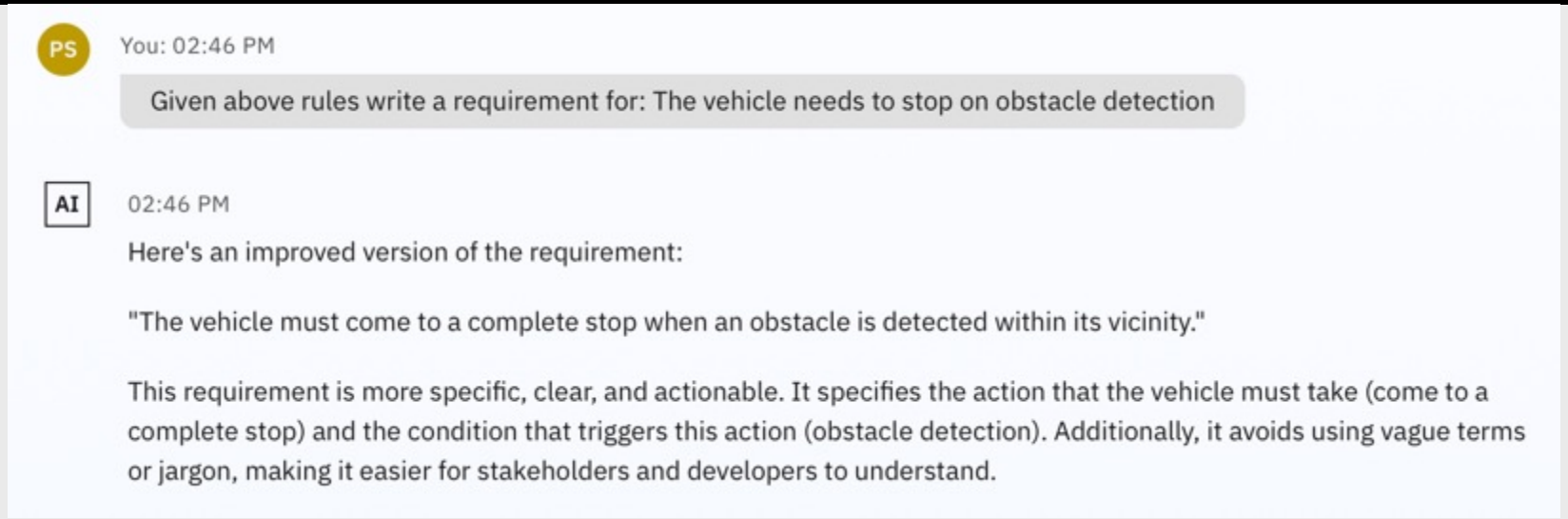
AI 02:43 PM

Writing clear, concise, and comprehensive requirements is crucial for the success of any project. Here are some best practices to consider when writing requirements:

1. Be specific: Use precise and unambiguous language to describe each requirement. Avoid vague terms and jargon. Instead, provide clear definitions or examples to ensure understanding.
2. Use user-centric language: Frame requirements from the perspective of the end-users or stakeholders. This helps ensure that the system meets their needs and expectations.
3. Prioritize requirements: Not all requirements are equally important. Rank them based on their importance and dependencies to help guide development and ensure that critical features are delivered first.
4. Keep requirements atomic: Break down complex requirements into smaller, manageable pieces. This makes it easier to understand, test, and validate each component independently.
5. Avoid unnecessary constraints: While it's important to define constraints, avoid limiting the solution space unnecessarily. Instead, focus on enabling the development team to create a flexible and adaptable system.
6. Validate requirements: Verify requirements with stakeholders and users to ensure that they accurately reflect their needs and expectations. Update requirements as needed based on feedback.
7. Traceability: Establish a traceability matrix to link requirements to design elements, test cases, and defects. This helps ensure that requirements are being fulfilled throughout the development lifecycle.
8. Review and update requirements: Regularly review and update requirements to ensure that they remain relevant and up-to-date. This helps prevent misunderstandings and ensures that the system continues to meet stakeholder needs.

By following these best practices, you can write clear and effective requirements that will guide your project to success.

# IBM watsonx – Write Good Requirements



# Requirements Challenges

Misunderstood requirements by stakeholders and engineers

Poorly expressed requirements → RQA: AI based quality checker

Misunderstanding or omission by development

Missed test coverage

Requirement change impact misunderstandings

Little reuse

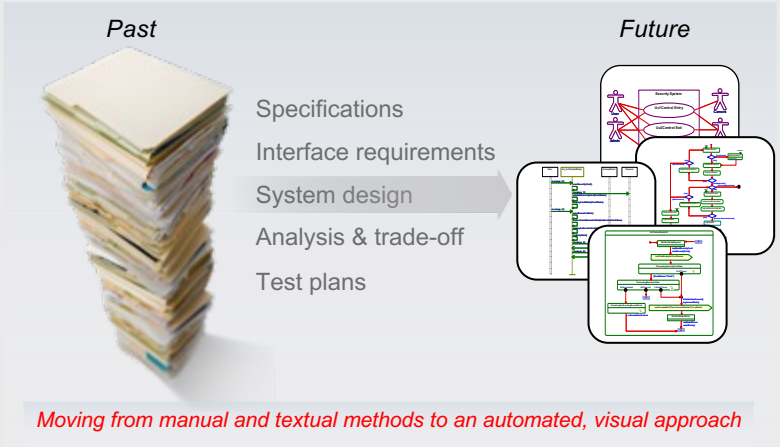
Complexity is the biggest challenge facing organizations today!

"Ok. That's what we need to do:
Thread A will pass event X to thread B and that will change B's state to Running from what it was before which was Idle. When B changes to Running it will send back an event Y to A and then wait for 2 second and then go back to Idle. Thread A will have started in Idle also and will go to Run after B sends back event Z which happens after the 2 seconds before going to Idle. All this should happen in less then 5 seconds."

# Modeling in Requirements Engineering

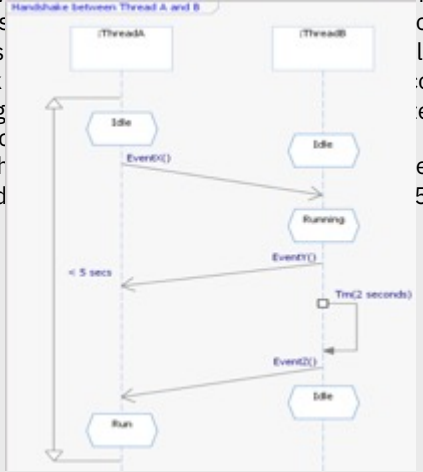
Requirements Engineering involves:

- Requirements elicitation
- Requirements analysis and negotiation
- **Requirements specification**
- **Functional analysis**
- **Requirements validation**
- Requirements management




*Moving from manual and textual methods to an automated, visual approach*

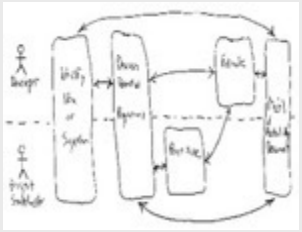
"Ok. That's what we need to do:
Thread A will pass event X to thread B and that will
change B's ore
which was ll
send back ond
and then g ed
in Idle alsc
event Z wh e
going to Id 5
seconds."



# Why Modeling?

Manage complexity

– Complicated applications need a visual plan



Simplify and abstract ! essential aspects of a system

– Increase understanding of requirements

Enhance communication

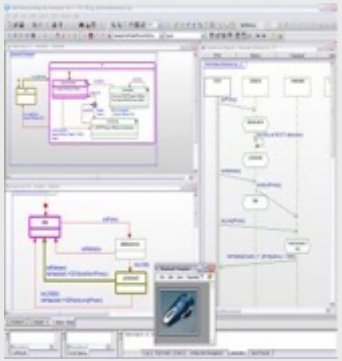
– Common language promotes common understanding across disciplines

Reduce risk

– Model execution increases knowledge and reduces uncertainty and risk

Provide traceability

– Models document what you have done



# Modeling in Requirements Engineering

Requirements Engineering involves:

- Requirements elicitation
- Requirements analysis and negotiation
- **Requirements specification – use cases and scenarios for describing user interactions**
- **Functional analysis – functional flows, interface definition, documented rationale**
- **Requirements validation – testing of requirements through model execution**
- Requirements management

2 Functional Requirements

2.1 Power car

2.1.1 Move car

2.1.1.1 Move forwards

The car shall be able to move forwards at all speeds from 0 to 200 kilometers per hour on standard flat roads with winds of 0 kilometers per hour, with 180 BHP.

2.1.1.2 Move backwards

The car shall be able to move backwards to a maximum speed of 20 Kilometers per hour on standard flat roads with winds of 0 kilometers per hour, with 180 BHP.

2.1.2 Accelerate car

The car shall be able to accelerate from 0 to 100 Kilometers per hour in 10 seconds on standard flat roads with winds of 0 kilometers per hour.

The car shall be able to accelerate from 100 to 150 kilometers per hour at a rate of 5 kilometers per second on standard flat roads with winds of 0 kilometers per hour.

The car shall be able to accelerate from 150 to 200 kilometers per hour at a rate of 3 kilometers per second on standard flat roads with winds of 0 kilometers per hour.

2.2 Control car

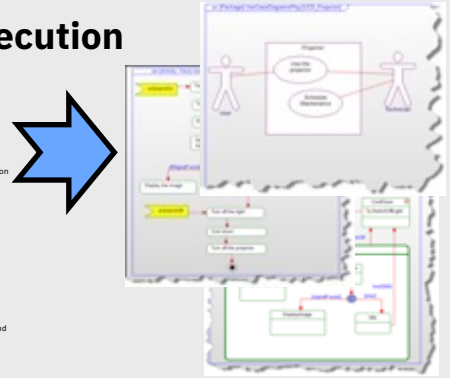
2.2.1 Switch on car

The car shall be able to discriminate which authorized people shall be able to switch on and operate the car.

2.2.2 Control speed

The car shall have a foot mechanism to control the speed of the car.

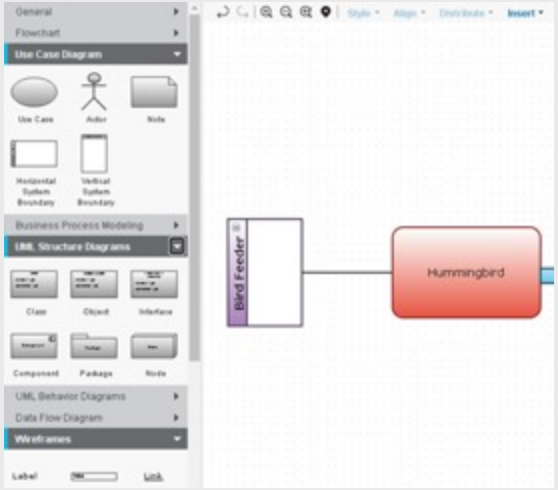
The speed control shall be infinitely variable from zero to maximum speed.



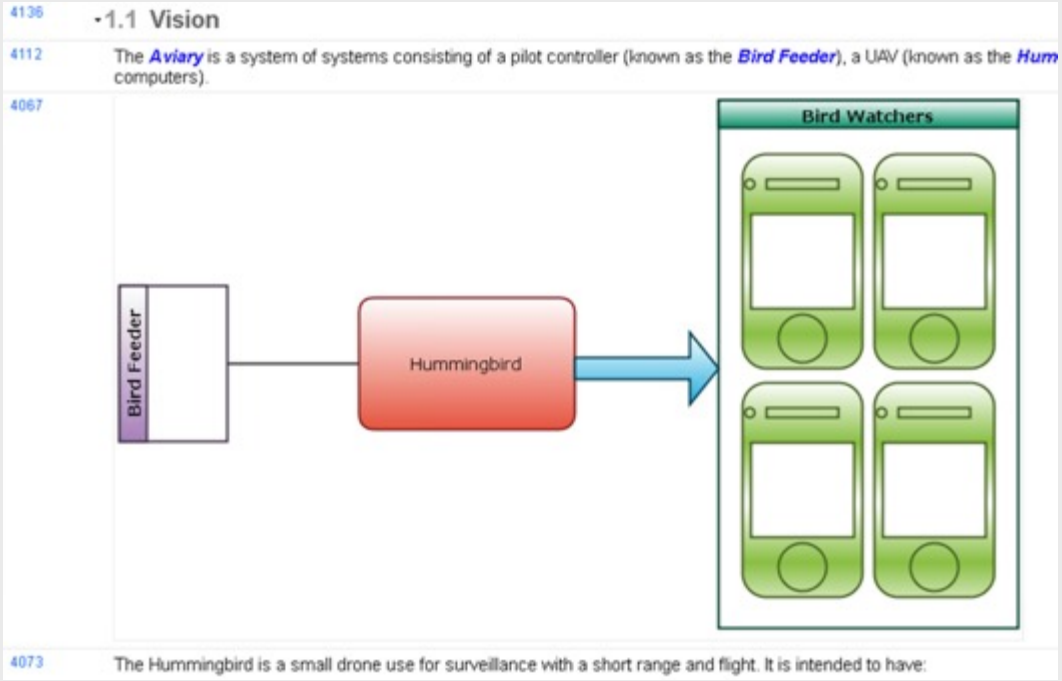
# Improve Requirements specification by just adding “Diagrams”

Fully integrated (like textual requirement)

Offers rich notations



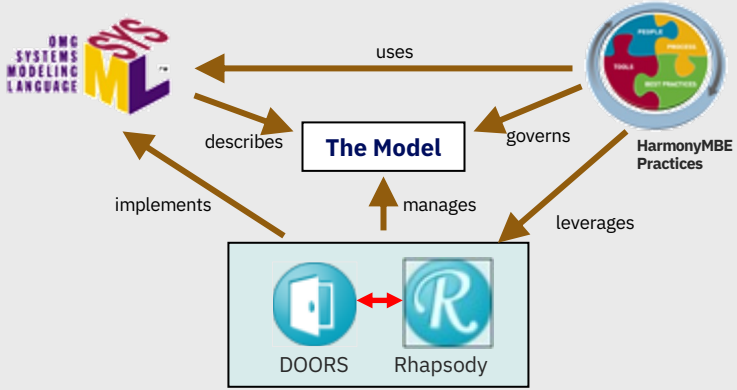

Not a formal model



# IBM’s Model Based Systems Engineering (MBSE) Solution

MBSE is a standards based Systems Engineering practice that incorporates:

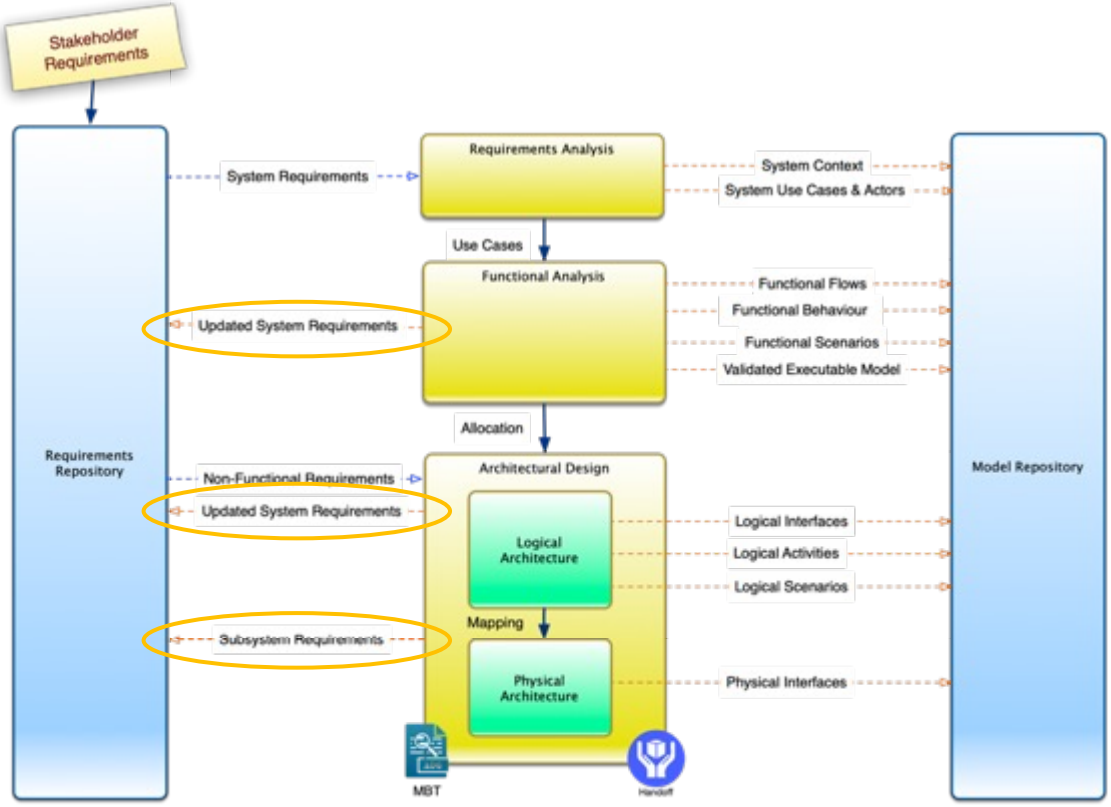
- Modeling language – SysML
- Modeling method – Harmony Systems Engineering Practices
- Modeling tool – Rhapsody for Systems Engineers & Rhapsody Model Manager
- Requirements management tool – DOORS Next Generation



# HarmonyMBE Best-Practice provides Guidance & Automation

Model Based Systems Engineering complements traditional requirements analysis techniques

- during Requirements Analysis, we organize requirements into functional groups (use cases)
- during Functional Analysis, we identify system functions and explore the system's dynamic behavior using activity diagrams and model execution
- during Architectural Design, operations are allocated to decomposed Logical Architecture
- Finally either direct Hand-Off to the engineering teams or first map to a Physical Architecture
- Missing, conflicting and derived Requirements are identified



IBM w

PS You: 02:47 PM

How will a system model be generated from requirements?

AI 02:47 PM

Generating a system model from requirements involves several steps. Here's a high-level overview of the process:

1. Analyze requirements: Begin by analyzing the requirements to identify the key components and relationships. Look for patterns, constraints, and dependencies between requirements.
2. Define the system architecture: Based on the requirements, define the overall system architecture. This includes identifying the main components, their interactions, and data flows.
3. Develop detailed designs: Create detailed designs for each component of the system. This includes specifying the algorithms, data structures, interfaces, and other implementation details.
4. Generate model elements: From the detailed designs, generate model elements such as class diagrams, sequence diagrams, and state diagrams. These models represent the structure, behavior, and constraints of the system.
5. Integrate model elements: Combine the model elements into a single, cohesive system model. This shows how the various components interact and how data flows through the system.
6. Validate the model: Validate the system model to ensure that it meets the requirements and that there are no inconsistencies or errors.
7. Refine the model: Based on the validation results, refine the system model as needed. This may involve making changes to the requirements, designs, or model elements.
8. Document the model: Document the system model in a clear and understandable manner. This includes providing descriptions, attributes, and relationships for each model element.

By following these steps, you can generate a system model that accurately represents the requirements and design of your system.

IBM

PS You: 02:50 PM

Define a system architecture for the requirement: The vehicle must come to a complete stop when an obstacle is detected within its vicinity.

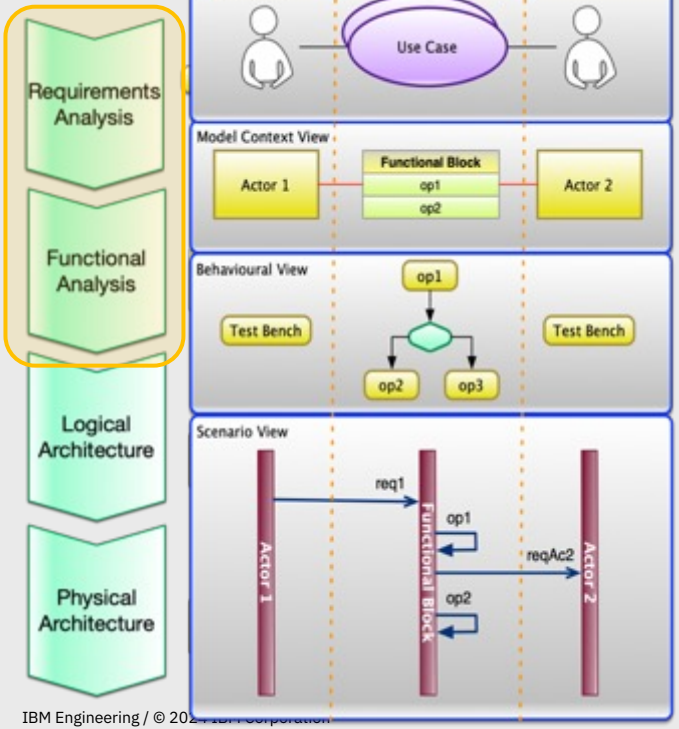
AI 02:50 PM

Here's a high-level system architecture for the given requirement:

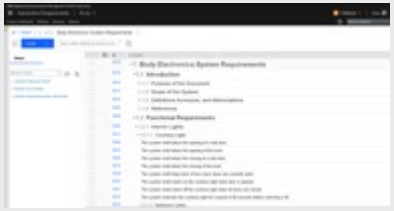
1. Sensor System: Responsible for detecting obstacles in the vehicle's vicinity. This could include cameras, lidar, ultrasonic sensors, or other appropriate technologies.
2. Obstacle Detection Module: Within the Sensor System, this module is responsible for processing sensor data and determining whether an obstacle has been detected.
3. Vehicle Control System: This system is responsible for controlling the vehicle's movement and stopping mechanisms. It receives input from the Obstacle Detection Module and takes appropriate actions to stop the vehicle.
4. Braking System: The braking system is responsible for bringing the vehicle to a complete stop. This could include hydraulic brakes, electric brakes, or other suitable braking technologies.
5. Communication System: This system enables communication between the Obstacle Detection Module, Vehicle Control System, and other vehicle systems. It could include CAN bus, Ethernet, or other appropriate communication protocols.

This system architecture provides a high-level overview of the components and interactions required to meet the requirement. However, it's important to note that the actual implementation may vary depending on the specific vehicle design, hardware, and software constraints.

# HarmonyMBE: Requirements Analysis & Functional Analysis: Results




Requirements allocated to Use Cases



System Context defined

Functional Flows defined

Test Scenarios captured

# Examples - How Modelling helps understanding and identifing Requirements

Shouldn’t actor X be involved in UC1 as well and how? What would be the IF?

Block2 needs input A but it’s not created anywhere

After seeing the scenario the stakeholder identified missing preconditions

Defining the system bahvior conflicting requirements were identified

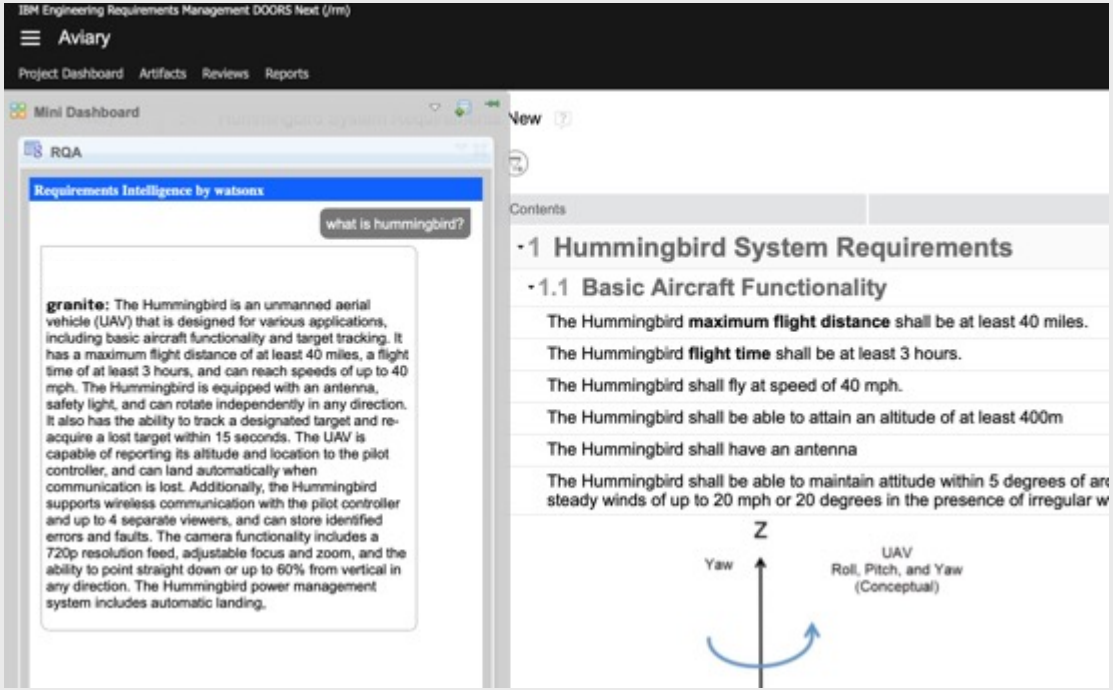
Analyzing the operational flow missing options were identified requiring addtl. Input

Executing the model unvieled wrong behavior based on unclear, missing requirements

…

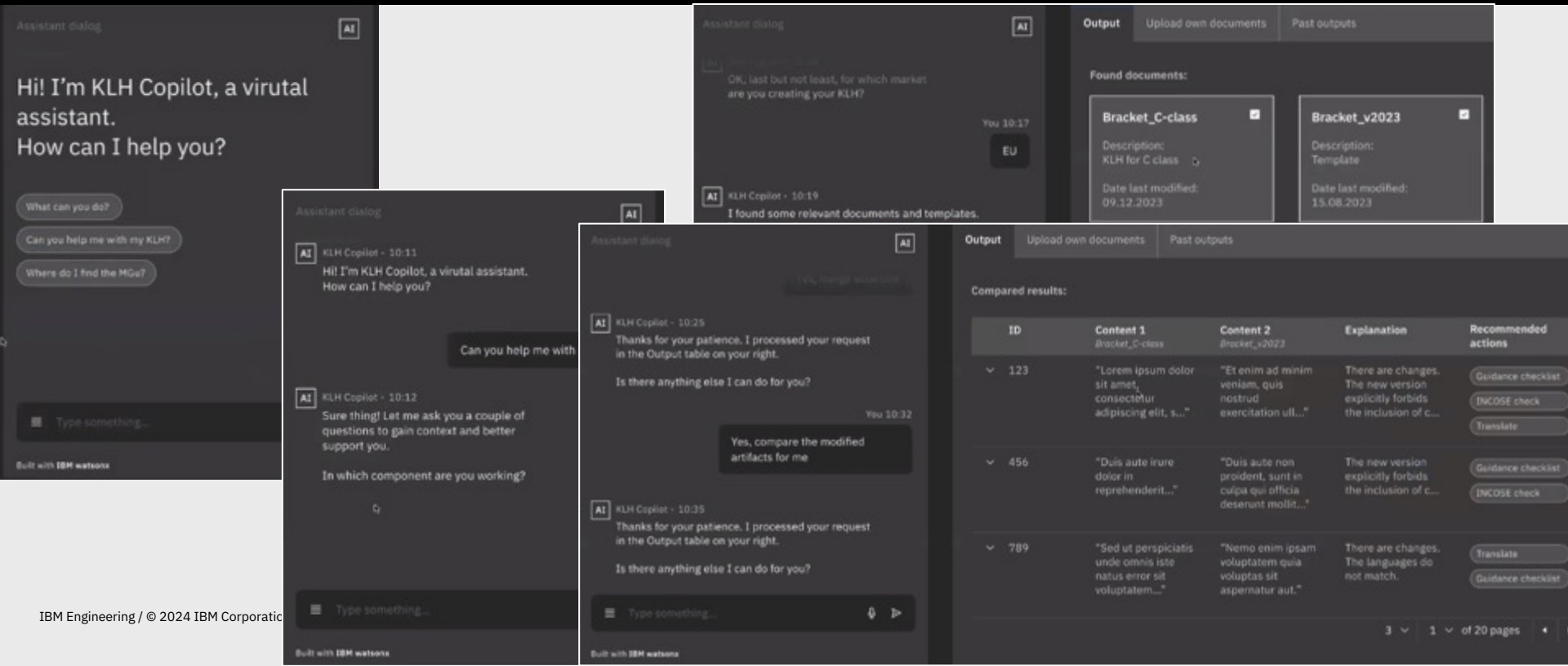
Certain diagrams become part of the spec.

# Examples - Can AI do the Job?




Stable Diffusion w prompt:
People looking at computer with requirements

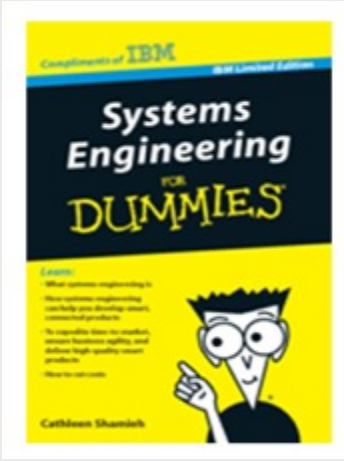
# Smart DOORS Next Copilot for Requirements Engineers PoC



# The Future ?

*GenAI alone cannot replace engineers, but engineers equipped with GenAI will surpass those without it.*

*vs.*



# Questions answered

- What is Requirements Engineering ?
  - Does AI help ? ✓
  - Can AI generate Requirements ? ✓
  - Does Modeling help ? ✓
- What is MBSE and how does it relate to Requirements Engineering ?
  - What is a MBSE best-practice? ✓
  - Can AI generate Models ? ✓
- Do Models generate Requirements ? ✓

# Thank You

Peter Schedl
peter.schedl@de.ibm.com

## Upcoming Events

Automotive Day Conference – München
TdSE – Leipzig

## Further information:

IBM Engineering Lifecycle Management Automotive Compliance
IBM Engineering Lifecycle Management Overview
IBM Engineering Lifecycle Management Interactive Tour





watsonx™