



Generative Artificial Intelligence: The Future of Model Based Systems Engineering

11 October 2023

Thomas Heckwolf, PMP
Patrick Buckley, PhD

About Me

Name: Thomas Heckwolf

Day Job: Sr. MBSE Engineer, i3

Supervisor: Dr. Patrick Buckley

Night Job: Ph.D. Student, Auburn University

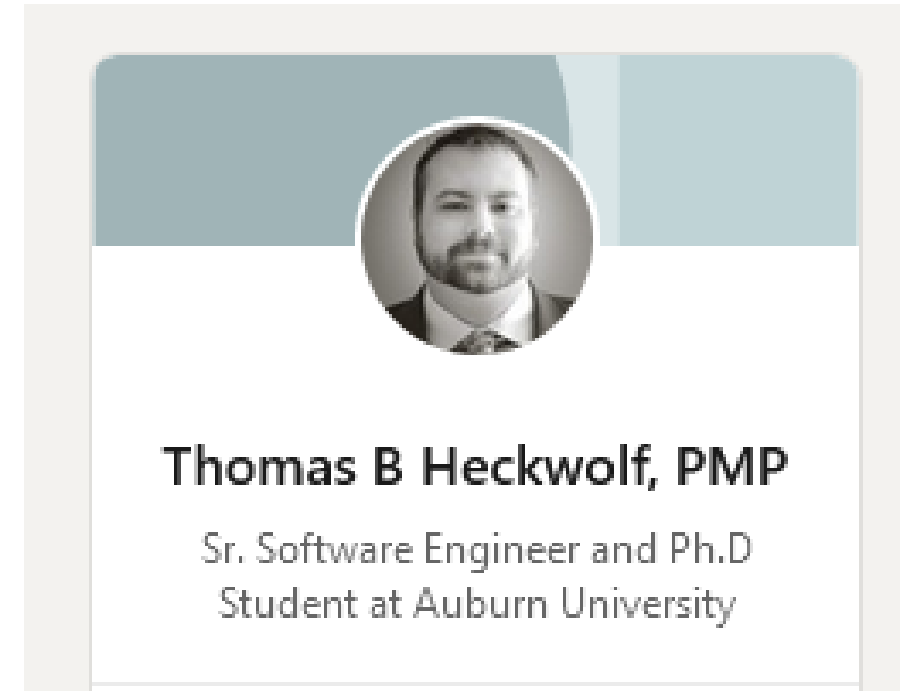
Research Interest: Software Engineering,
Blockchain, Software Architecture,
Artificial Intelligence

Advisor: Dr. Gerry Dozier

Email: Thomas.Heckwolf@i3-corps.com

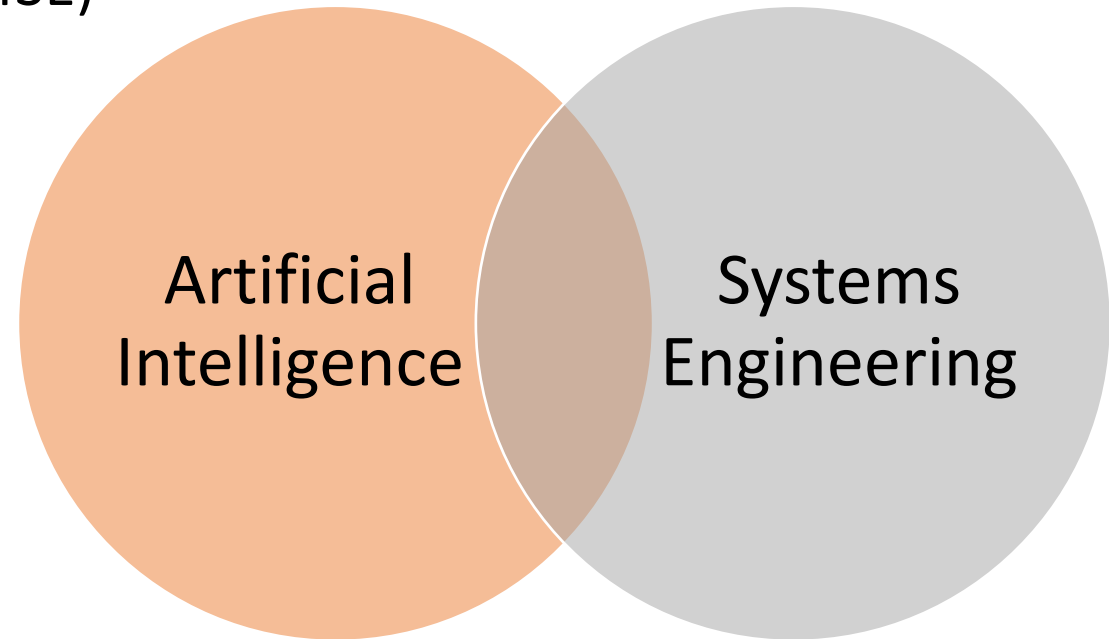


AUBURN UNIVERSITY
ARTIFICIAL INTELLIGENCE INITIATIVE

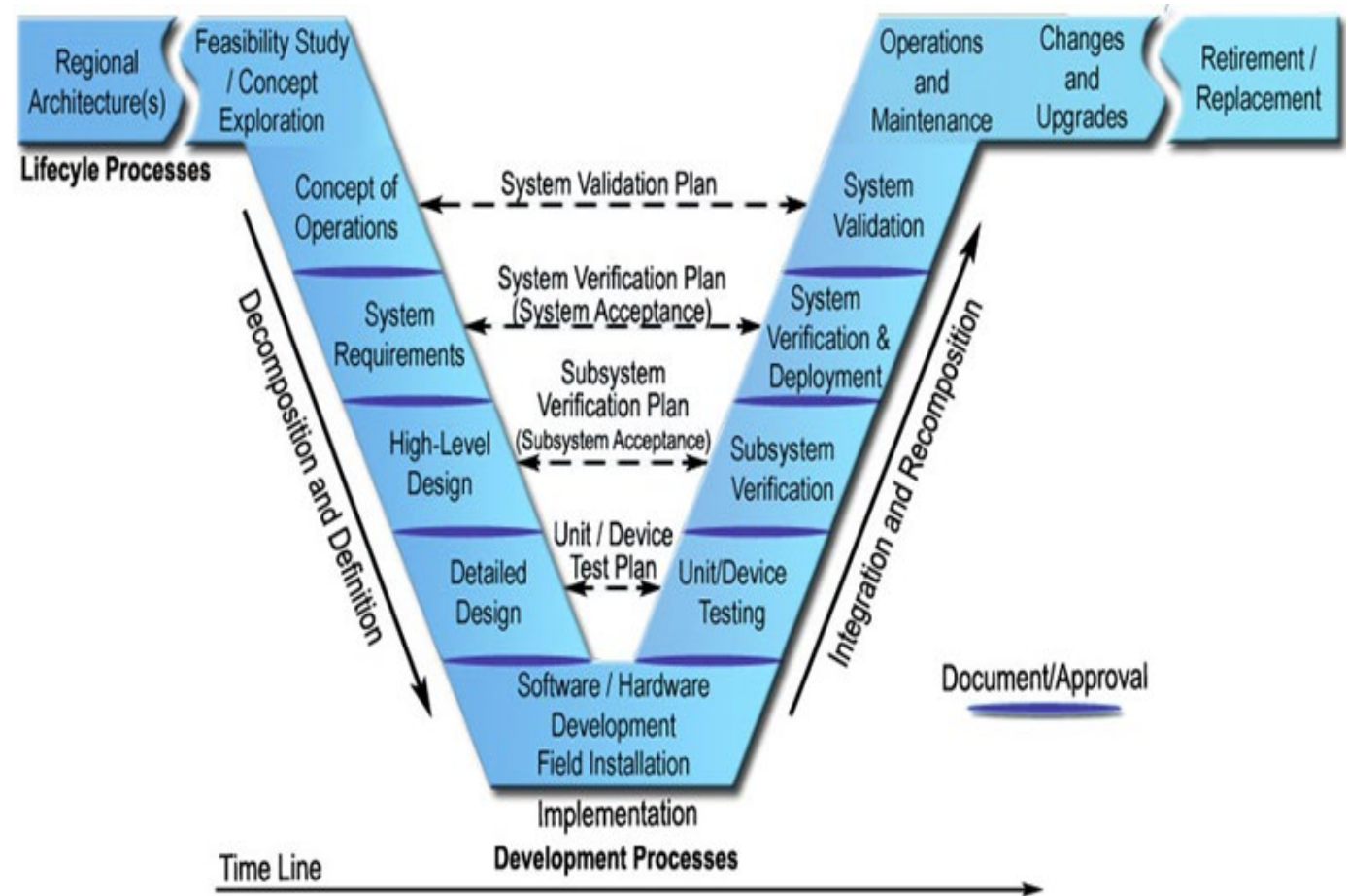
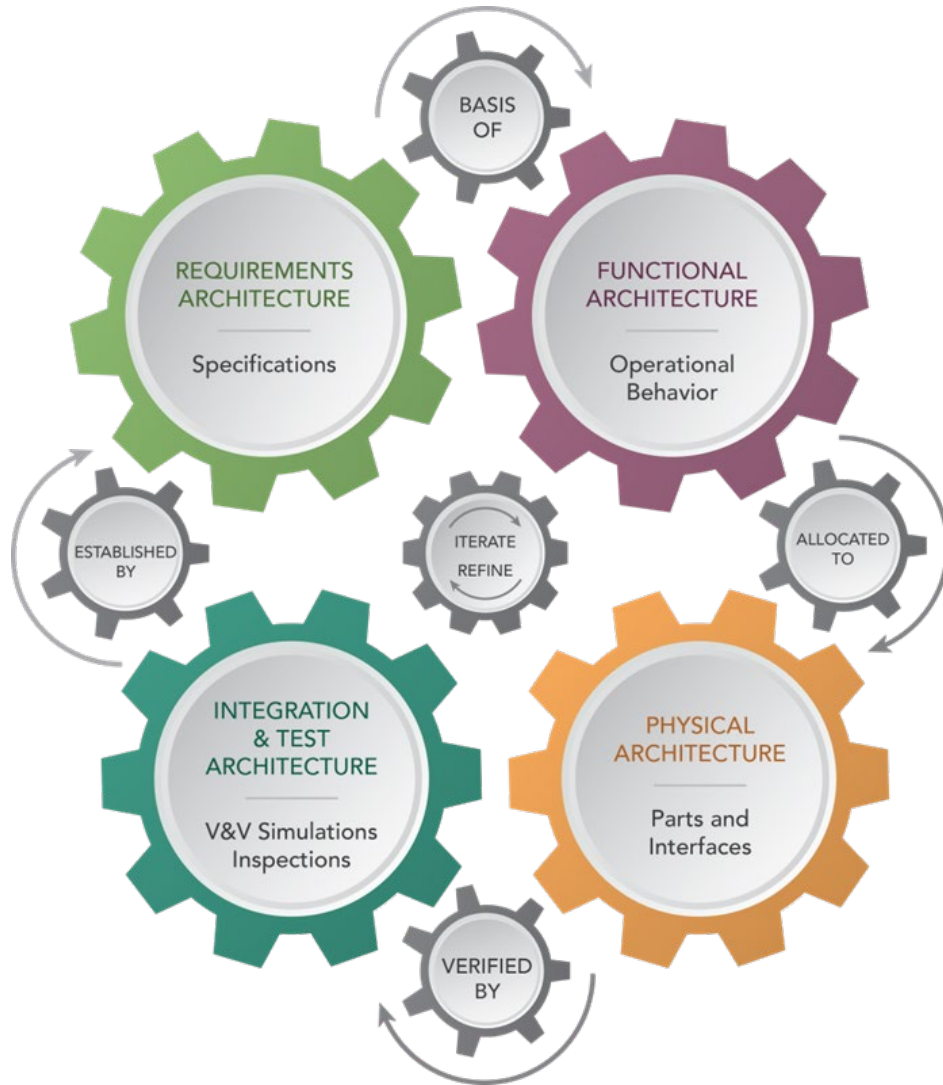


Why Are We Here

- Talk about Artificial Intelligence (AI) and Model Based Systems Engineering
- Give a brief overview on AI
- Talk about AI for Systems Engineering (AI4SE)
- Discuss Generative AI
- Talk about ways to use Generative AI



What is Model Based Systems Engineering (MBSE)



Artificial Intelligence / Machine Learning

- **What is Artificial Intelligence?**

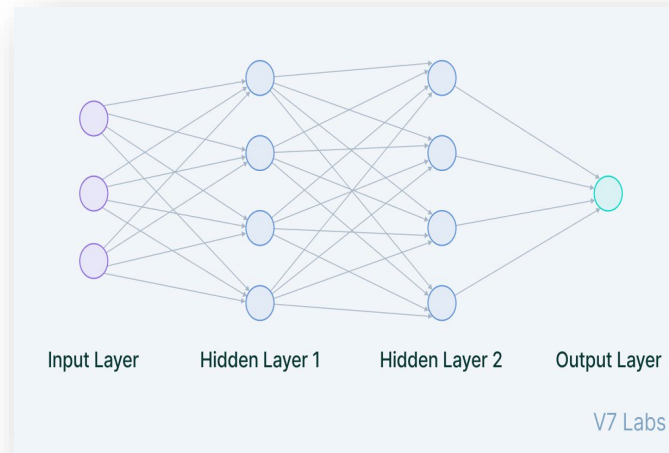
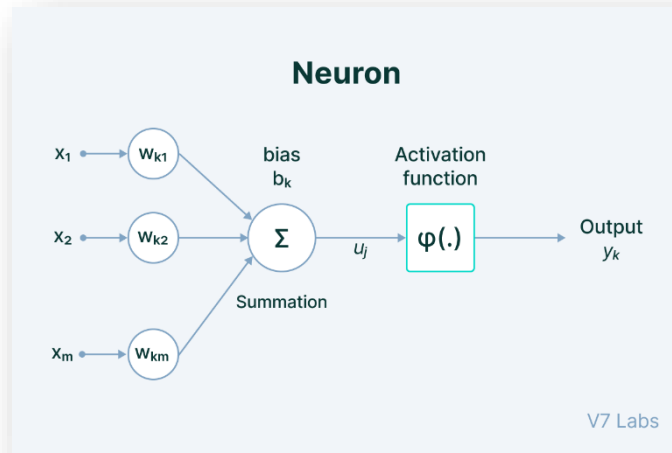
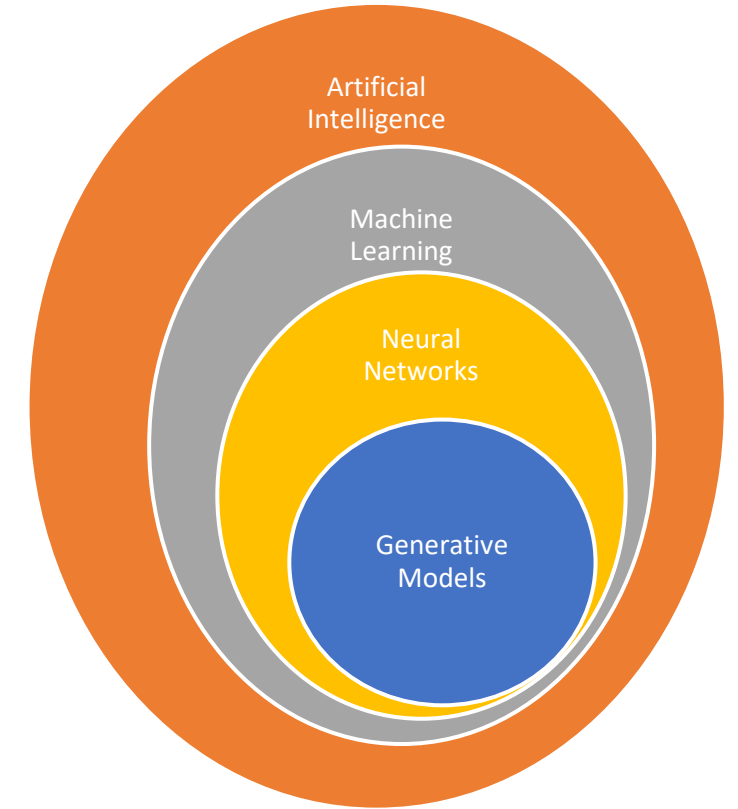
- “It is the science and engineering of making intelligent machines, especially intelligent computer programs.” - McCarthy

- **What is Machine Learning?**

- “Machine learning is a branch of artificial intelligence and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy.” – IBM

- **What are Neural Networks?**

- “Neural networks, also known as artificial neural networks (ANNs) or simulated neural networks (SNNs), are a subset of machine learning” - IBM



AI for Systems Engineering

- **AI4SE:**

- The goal of AI4SE is to improve the efficiency, effectiveness, and quality of software engineering processes, by leveraging the capabilities of AI and machine learning to automate, augment, and enhance various aspects of software development, testing, and maintenance. - Ilma

1. **Code generation and manipulation**
2. Code analysis and diagnosis
3. Testing and verification
4. Project management and planning
5. **Requirements analysis and specification**
6. Collaborative systems

Generative AI in Systems Engineering – AI4SE

- **Pros**

- Optimizing System Engineer's Workflow
- Optimizing Design Processes
- Enhancing Simulation and Analysis
- Accelerating Prototyping and Manufacturing

- **Cons**

- Bias and Fairness
- Data Privacy and Security
- Lack of Transparency and Interpretability
- Resource Intensiveness
- Ethical Considerations



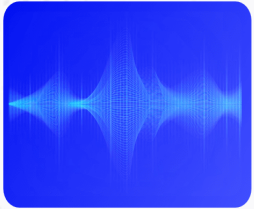
Generative Artificial Intelligence

webclues

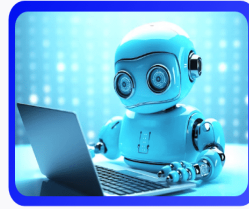
Type your prompt here

Generative Artificial Intelligence

Audio



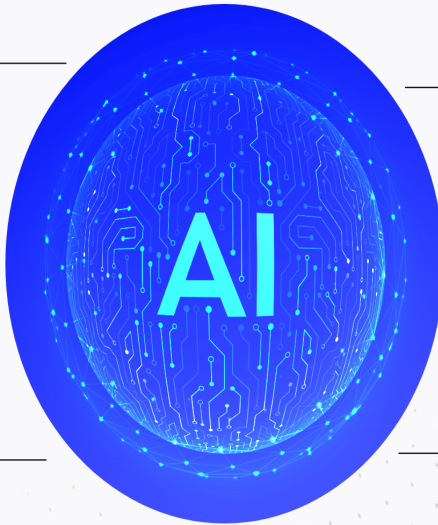
Images



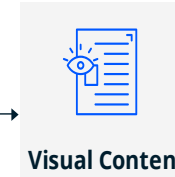
Video



Text



Generative AI use cases



Visual Content

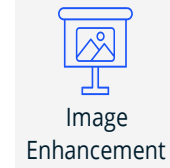
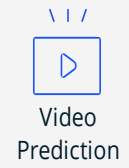
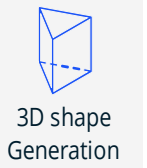


Image Enhancement



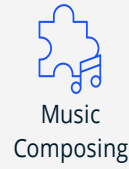
Video Prediction



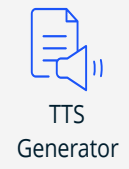
3D shape Generation



Audio Generation



Music Composing



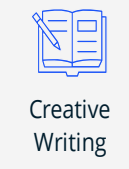
TTS Generator



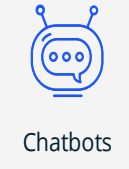
STS Conversion



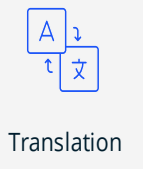
Text Generation



Creative Writing



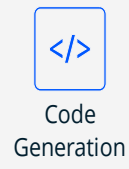
Chatbots



Translation



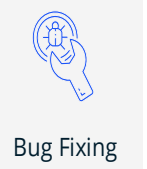
Code Generation



Code Generation



Code compilation



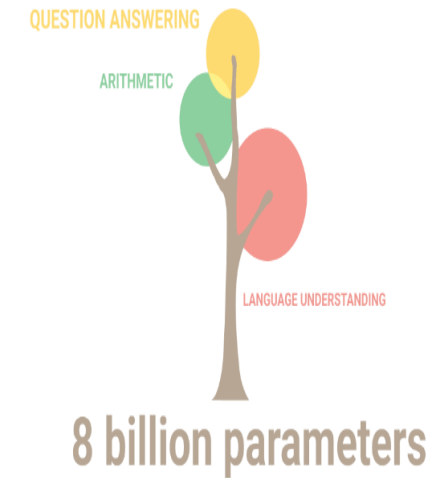
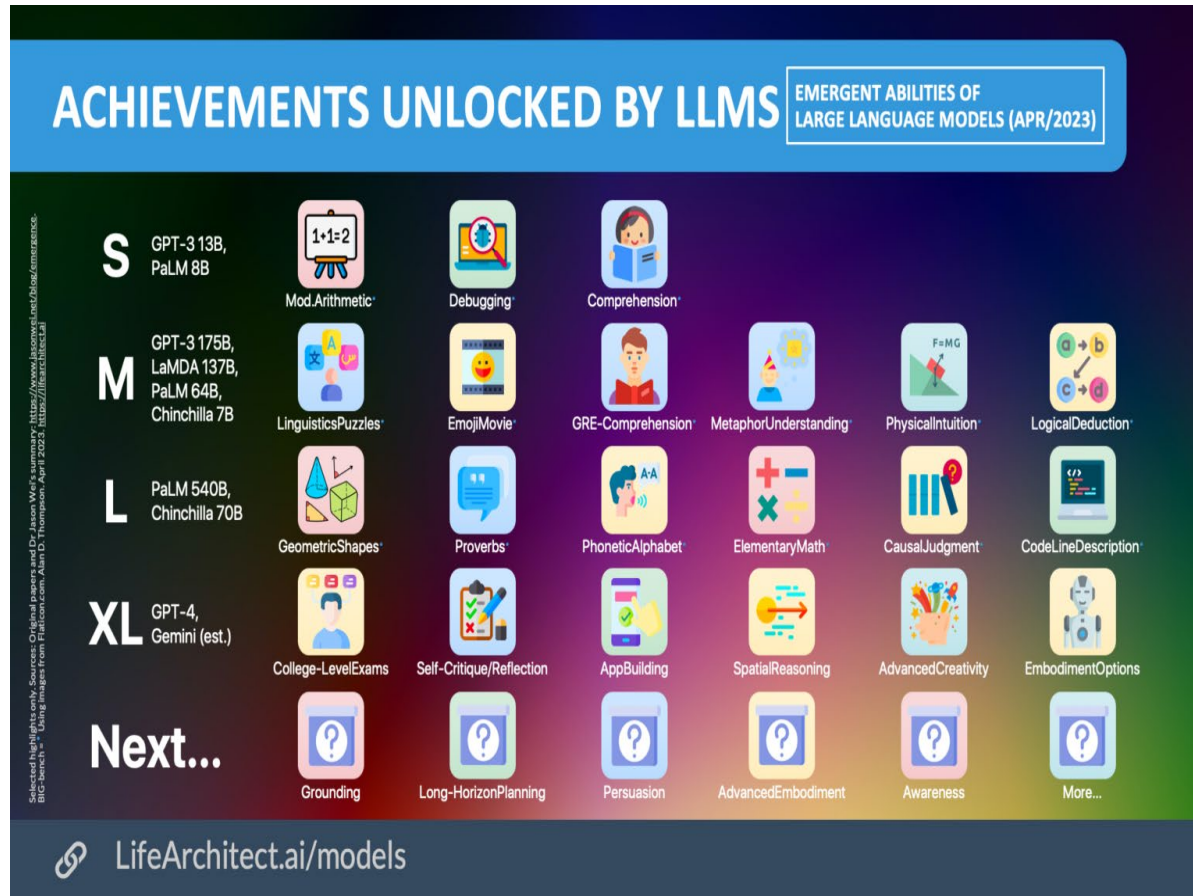
Bug Fixing

LeewayHertz



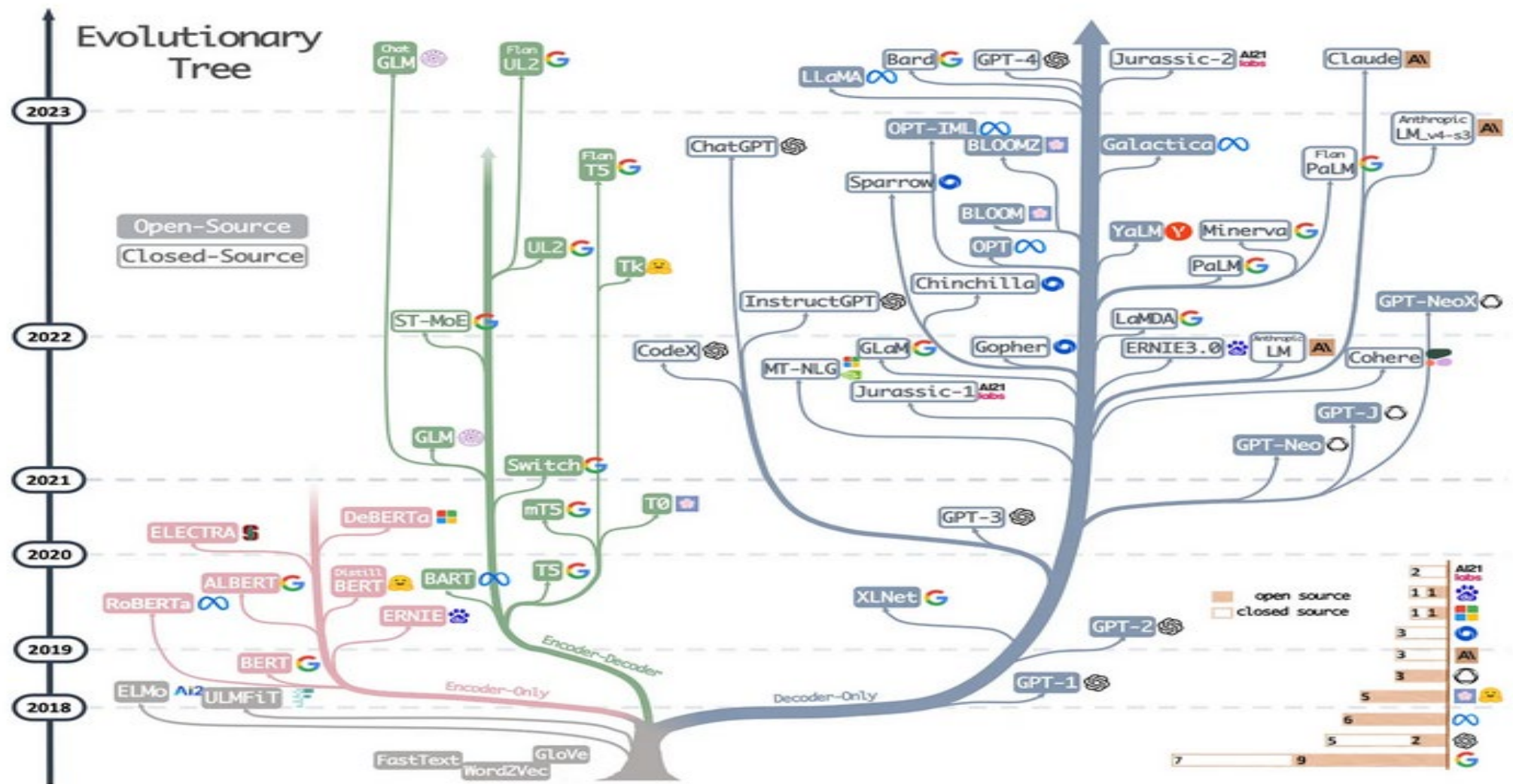
INFORM. INSPIRE. IMPROVE.

LLM Capabilities Based on Data



<https://ai.googleblog.com/2022/04/pathways-language-model-palm-scaling-to.html?ref=assemblyai.com>

Evolution with Generative AI



<https://arxiv.org/abs/2304.13712>



Ways to Use Generative AI: Requirements



Automating Requirements Generation

- **Requirement Elicitation**

- Automate the elicitation of requirements from stakeholders
- Learn from past projects and suggest requirements based on similar projects

- **Requirement Analysis and Validation**

- Analyze and validate requirements
- Identify ambiguous or conflicting requirements
- Validate requirements meet certain criteria or
- Suggest changes to improve the clarity or completeness of the requirements

- **Requirement Management and Maintenance**

- Help with the ongoing management and maintenance of requirements
- Track changes to requirements
- Manage dependencies between requirements
- Predict the impact of requirement changes on the project

Automating Requirements Generation - Drawbacks

- **Dependence on Data Quality**
 - Only as good as the data they are trained on
- **Interpretation of Requirements**
 - Doesn't truly understand it in the way humans do
- **Lack of Creativity and Innovation**
 - Can struggle with tasks requiring creative thinking or novel solutions
- **Difficulty Handling Ambiguity**
 - Ambiguity is common in requirements elicitation and often requires human judgement to resolve
- **Trust and Acceptance**
 - Stakeholders may be reluctant to trust or accept requirements generated by an AI



Requirement Elicitation: Generate 5 Requirements for a Login System and 5 Derived Requirements – “GPT-4”

Let's consider a simple login system for a hypothetical web application. Here are five primary requirements:

Primary Requirements:

1. **R1:** The system shall allow registered users to authenticate using a username and password.
2. **R2:** The system shall lock a user's account after three consecutive failed login attempts.
3. **R3:** The system shall provide a "Forgot Password" function for users who cannot remember their password.
4. **R4:** The system shall encrypt user passwords during transmission and storage.
5. **R5:** The system shall log all successful and unsuccessful login attempts.

Now let's derive some requirements from these primary ones:

Derived Requirements:

1. **DR1 (from R1):** The system shall provide an error message when a user enters an incorrect username or password.
2. **DR2 (from R2):** The system shall provide a mechanism for unlocking a locked account. This process should include identity verification steps.
3. **DR3 (from R3):** The "Forgot Password" function shall generate a secure, temporary link that is sent to the user's registered email address.
4. **DR4 (from R4):** The system shall use a secure encryption algorithm, such as BCrypt or Argon2, for password encryption.
5. **DR5 (from R5):** The system shall include the date, time, and IP address in its log of login attempts.

These requirements and derived requirements provide a basic outline for the login system. Each requirement should be reviewed and validated to ensure it meets the needs of the users and the security standards of the organization.



Requirement Analysis and Validation: Analysis, Validate Requirements Meet EARS and Update to EARS Generated Requirements – “GPT-4”

EARS (Easy Approach to Requirements Syntax) is a method for writing software requirements in a clear and structured way. Each EARS requirement is written as a simple sentence with a specific structure, which makes it easy to understand and validate.

Let's take the primary requirements from the previous response and re-write them using EARS:

Primary Requirements (in EARS syntax):

1. **R1:** When the user attempts to log in, the system should authenticate the user if the correct username and password are provided.
2. **R2:** When the user has failed to log in three times in a row, the system should lock the user's account.
3. **R3:** When the user selects the "Forgot Password" option, the system should provide a mechanism to reset the user's password.
4. **R4:** While the system is transmitting or storing user passwords, it should encrypt them.
5. **R5:** Whenever a login attempt is made, the system should log the attempt, whether it was successful or not.

And here are the derived requirements, also re-written using EARS:

Derived Requirements (in EARS syntax):

1. **DR1:** When the user enters an incorrect username or password, the system should display an error message.
2. **DR2:** When a user's account is locked due to failed logins, the system should provide a mechanism for unlocking the account after identity verification.
3. **DR3:** When the user initiates the "Forgot Password" process, the system should generate a secure, temporary link and send it to the user's registered email address.
4. **DR4:** While the system is encrypting user passwords, it should utilize a secure encryption algorithm such as BCrypt or Argon2.
5. **DR5:** When the system logs a login attempt, it should include the date, time, and IP address.



Requirement Analysis and Validation: Verify No Vague or Redundant Requirements and Add Parametrics to Requirements – “GPT-4”

Upon review, the requirements seem to be clear and free of redundancy. Each requirement focuses on a unique aspect of the login system. To make these requirements more precise and measurable, let's add some parametrics:

Primary Requirements with Parametrics:

1. **R1:** When the user attempts to log in, the system should authenticate the user within 2 seconds if the correct username and password are provided.
2. **R2:** When the user has failed to log in three times in a row within 15 minutes, the system should lock the user's account for a minimum of 30 minutes.
3. **R3:** When the user selects the "Forgot Password" option, the system should send the password reset link to the user's registered email within 5 minutes.
4. **R4:** While the system is transmitting or storing user passwords, it should encrypt them using a method that meets or exceeds the current industry standards.
5. **R5:** Whenever a login attempt is made, the system should log the attempt within 1 second, whether it was successful or not.

Derived Requirements with Parametrics:

1. **DR1:** When the user enters an incorrect username or password, the system should display an error message within 1 second.
2. **DR2:** When a user's account is locked due to failed logins, the system should provide a mechanism for unlocking the account after identity verification within 24 hours.
3. **DR3:** When the user initiates the "Forgot Password" process, the system should generate a secure, temporary link valid for 1 hour and send it to the user's registered email address.
4. **DR4:** While the system is encrypting user passwords, it should utilize a secure encryption algorithm such as BCrypt or Argon2 that meets or exceeds the current industry standards.
5. **DR5:** When the system logs a login attempt, it should include the date, time, and IP address in the log record, and this information should be retrievable within 1 second for the current date.

Adding parametrics to these requirements makes them more specific, measurable, and testable, which will help to ensure that the system meets the intended performance and functionality standards.

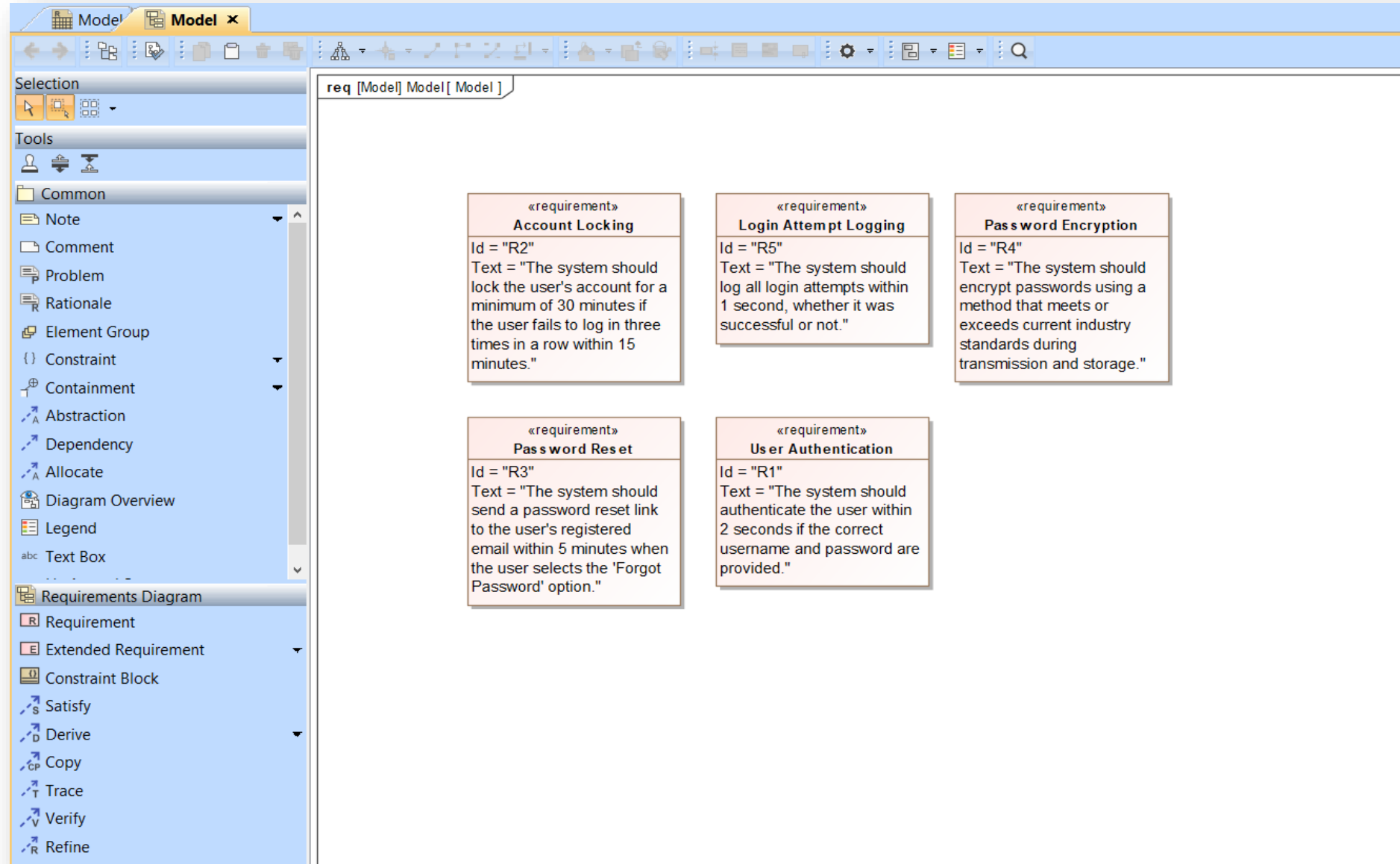
Requirement Management and Maintenance: Track Changes from Original Generated Changes to New Requirements – “GPT-4”

Let's demonstrate this process manually by showing the progression of our requirements from their original form to their current (EARS and parametric) form:

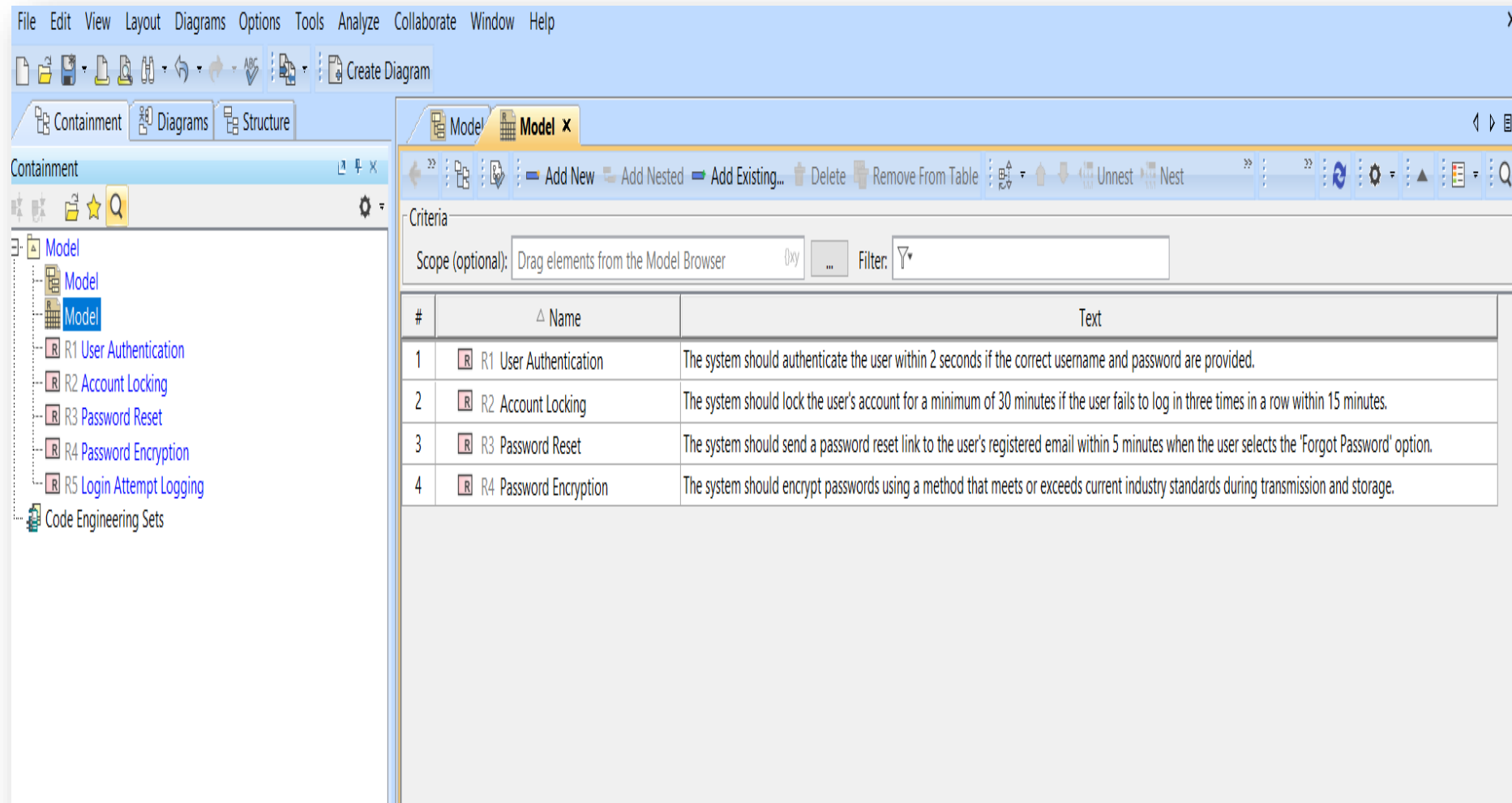
1. **R1 Original:** The system shall allow registered users to authenticate using a username and password.
 1. **R1 EARS:** When the user attempts to log in, the system should authenticate the user if the correct username and password are provided.
 2. **R1 Parametric:** When the user attempts to log in, the system should authenticate the user within 2 seconds if the correct username and password are provided.
2. **R2 Original:** The system shall lock a user's account after three consecutive failed login attempts.
 1. **R2 EARS:** When the user has failed to log in three times in a row, the system should lock the user's account.
 2. **R2 Parametric:** When the user has failed to log in three times in a row within 15 minutes, the system should lock the user's account for a minimum of 30 minutes.
3. **R3 Original:** The system shall provide a "Forgot Password" function for users who cannot remember their password.
 1. **R3 EARS:** When the user selects the "Forgot Password" option, the system should provide a mechanism to reset the user's password.
 2. **R3 Parametric:** When the user selects the "Forgot Password" option, the system should send the password reset link to the user's registered email within 5 minutes.
4. **R4 Original:** The system shall encrypt user passwords during transmission and storage.
 1. **R4 EARS:** While the system is transmitting or storing user passwords, it should encrypt them.
 2. **R4 Parametric:** While the system is transmitting or storing user passwords, it should encrypt them using a method that meets or exceeds the current industry standards.
5. **R5 Original:** The system shall log all successful and unsuccessful login attempts.
 1. **R5 EARS:** Whenever a login attempt is made, the system should log the attempt, whether it was successful or not.
 2. **R5 Parametric:** Whenever a login attempt is made, the system should log the attempt within 1 second, whether it was successful or not.



Requirements Ingested into Cameo EA – Requirements Model



Requirements Ingested into Cameo EA – Requirements Table



The screenshot displays the Cameo EA software interface. On the left, the 'Containment' pane shows a hierarchical tree with 'Model' at the top, followed by 'Model' (selected), and then a list of requirements: 'R1 User Authentication', 'R2 Account Locking', 'R3 Password Reset', 'R4 Password Encryption', and 'R5 Login Attempt Logging'. Below these is 'Code Engineering Sets'. The main area on the right is the 'Requirements Table'. It has a toolbar with buttons like 'Add New', 'Add Nested', 'Add Existing...', 'Delete', 'Remove From Table', 'Unnest', and 'Nest'. Below the toolbar is a 'Criteria' section with a 'Scope (optional):' dropdown set to 'Drag elements from the Model Browser' and a 'Filter:' dropdown. The table itself has three columns: '#', '△ Name', and 'Text'. It contains four rows of requirements.

#	△ Name	Text
1	R1 User Authentication	The system should authenticate the user within 2 seconds if the correct username and password are provided.
2	R2 Account Locking	The system should lock the user's account for a minimum of 30 minutes if the user fails to log in three times in a row within 15 minutes.
3	R3 Password Reset	The system should send a password reset link to the user's registered email within 5 minutes when the user selects the 'Forgot Password' option.
4	R4 Password Encryption	The system should encrypt passwords using a method that meets or exceeds current industry standards during transmission and storage.

Ways to Use Generative AI: Use Cases



Use Case Generation: Generate Actors and Use Cases from the Requirements – “GPT-4”

Let's identify the actors and create use cases based on the parametric requirements that we have developed so far.

Actors:

- User:** The person who interacts with the login system. This includes legitimate users who are trying to log in, as well as unauthorized users who might be attempting to break in.
- System:** The login system itself, which processes login attempts and responds according to the defined requirements.

Use Cases:

1. UC1 - User Authentication

1. **Actor:** User
2. **Precondition:** User is on the login page.
3. **Flow:** The user enters their username and password. The system checks the credentials and, if they are correct, logs the user in within 2 seconds.
4. **Postcondition:** User is logged in and redirected to the main page of the application.

2. UC2 - Account Locking

1. **Actor:** System
2. **Precondition:** User has failed to log in three times in a row within 15 minutes.
3. **Flow:** The system automatically locks the user's account for a minimum of 30 minutes.
4. **Postcondition:** User is unable to log in until the account is unlocked.

3. UC3 - Password Reset

1. **Actor:** User
2. **Precondition:** User has forgotten their password.
3. **Flow:** User clicks "Forgot Password," and the system sends a reset link to their registered email within 5 minutes.
4. **Postcondition:** User is able to reset their password using the link received.

4. UC4 - Password Encryption

1. **Actor:** System
2. **Precondition:** User enters a password (either during login or password reset).
3. **Flow:** System encrypts the password during transmission and storage using a method that meets or exceeds current industry standards.
4. **Postcondition:** User's password is securely stored and transmitted.

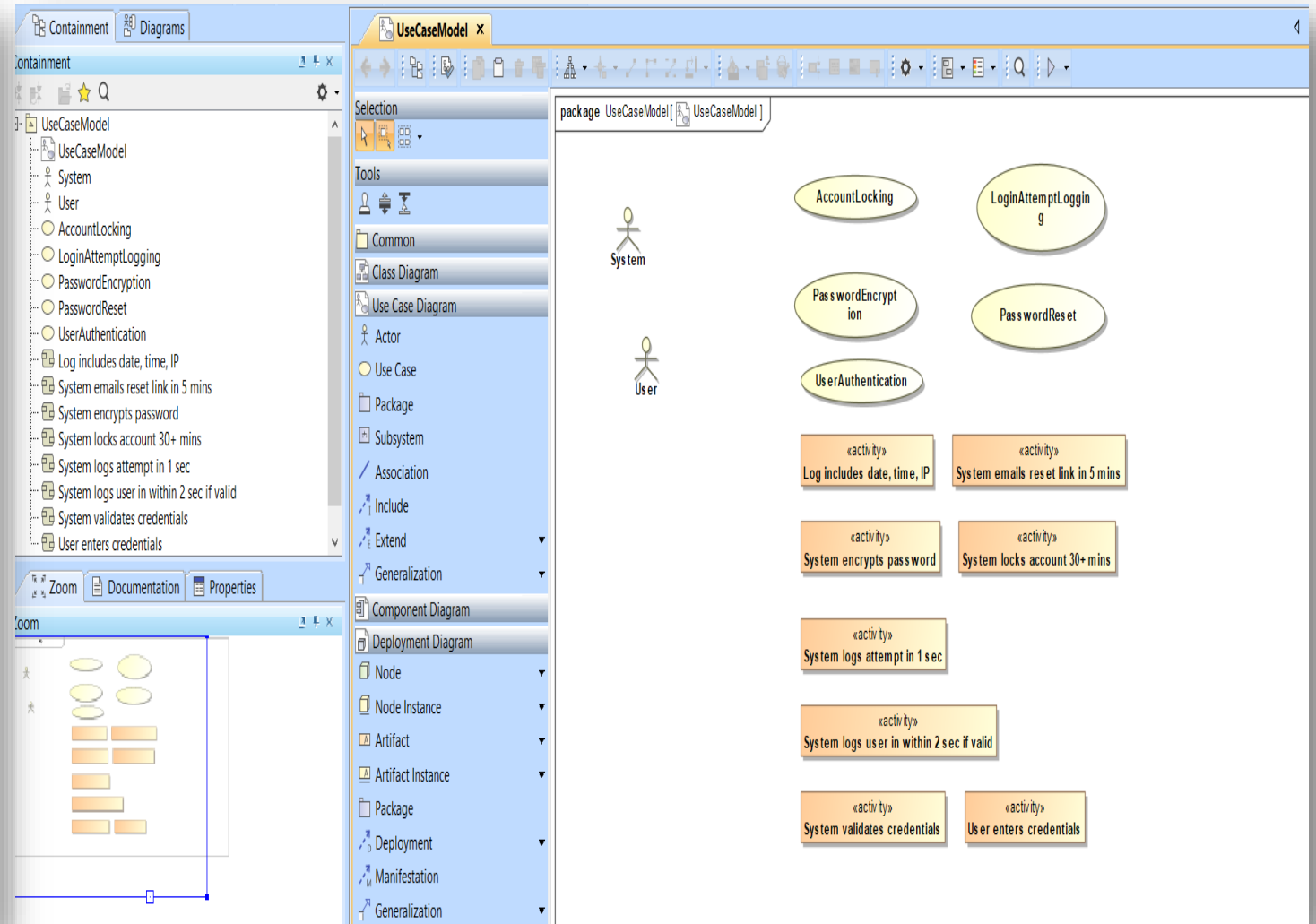
5. UC5 - Login Attempt Logging

1. **Actor:** System
2. **Precondition:** A login attempt is made (successful or unsuccessful).
3. **Flow:** The system logs the attempt within 1 second, including the date, time, and IP address.
4. **Postcondition:** The login attempt information is stored and can be retrieved for review.



Use Case Generation: Generate Actors and Use Cases from the Requirements XMI – “Claude”

```
xml
Copy
<?xml version="1.0" encoding="UTF-8"?>
<xmi:XMI xmi:version="2.1" xmlns:xmi="http://schema.omg.org/spec/XMI/2.1" xmlns:uml="http://www.eclipse.org/uml2/5.0.0/UML">
  <uml:Model xmi:id="_18_0_62a02f_1504452906660_678460_8649" name="UseCaseModel">
    <packagedElement xmi:type="uml:Actor" xmi:id="_18_0_62a02f_1504452906666_31407_8650" name="User"/>
    <packagedElement xmi:type="uml:Actor" xmi:id="_18_0_62a02f_1504452906667_297024_8651" name="System"/>
    <packagedElement xmi:type="uml:UseCase" xmi:id="_18_0_62a02f_1504452906668_456566_8652" name="UserAuthentication">
      <include xmi:id="_18_0_62a02f_1504452906669_562310_8653" addition="_18_0_62a02f_1504452906670_518358_8654"/>
      <include xmi:id="_18_0_62a02f_1504452906671_673480_8655" addition="_18_0_62a02f_1504452906672_250022_8656"/>
      <include xmi:id="_18_0_62a02f_1504452906673_994808_8657" addition="_18_0_62a02f_1504452906674_861364_8658"/>
    </packagedElement>
    <packagedElement xmi:type="uml:UseCase" xmi:id="_18_0_62a02f_1504452906675_575302_8659" name="AccountLocking">
      <include xmi:id="_18_0_62a02f_1504452906676_432778_8660" addition="_18_0_62a02f_1504452906677_223220_8661"/>
    </packagedElement>
    <packagedElement xmi:type="uml:UseCase" xmi:id="_18_0_62a02f_1504452906678_454754_8662" name="PasswordReset">
      <include xmi:id="_18_0_62a02f_1504452906679_461564_8663" addition="_18_0_62a02f_1504452906680_648274_8664"/>
    </packagedElement>
    <packagedElement xmi:type="uml:UseCase" xmi:id="_18_0_62a02f_1504452906681_648274_8664" name="UserAuthentication">
      <include xmi:id="_18_0_62a02f_1504452906682_648274_8664" addition="_18_0_62a02f_1504452906683_648274_8664"/>
    </packagedElement>
  </uml:Model>
</xmi:XMI>
```



Ways to Use Generative AI: Activity Generation



Activity Generation: Generate Recommended Activity Diagrams – “Bard”

Activity Diagram for User Authentication (UC1)

Start

- > User enters username and password
- > System checks credentials
- > If credentials are correct:
 - > System logs user in
 - > System redirects user to main page
- > Else:
 - > System displays error message

End

Activity Diagram for Account Locking (UC2)

Start

- > System checks if user has failed to log in three times in a row within 15 minutes
- > If yes:
 - > System locks user's account for 30 minutes
 - > System displays message to user informing them that their account has been locked
- > Else:
 - > System does nothing

End

Activity Generation: Generate Activity Diagram Nodes XMI – “Claude”

xml

Copy

```
<?xml version="1.0" encoding="UTF-8"?>
<xmi:XMI xmlns:xmi="http://schema.omg.org/spec/XMI/2.1" xmlns:uml="http://www.eclipse.org/uml2/5.0.0/UML">

  <uml:Model name="ActivityModels">

    <packagedElement xmi:type="uml:Activity" name="UserLogin">

      <node xmi:type="uml:InitialNode" name="Start" />

      <node xmi:type="uml:AcceptEventAction" name="WaitForAuthentic
ationRequest"/>

      <node xmi:type="uml:OpaqueAction" name="GetCredentials"/>

      <node xmi:type="uml:OpaqueAction" name="ValidateCredentials"/
>

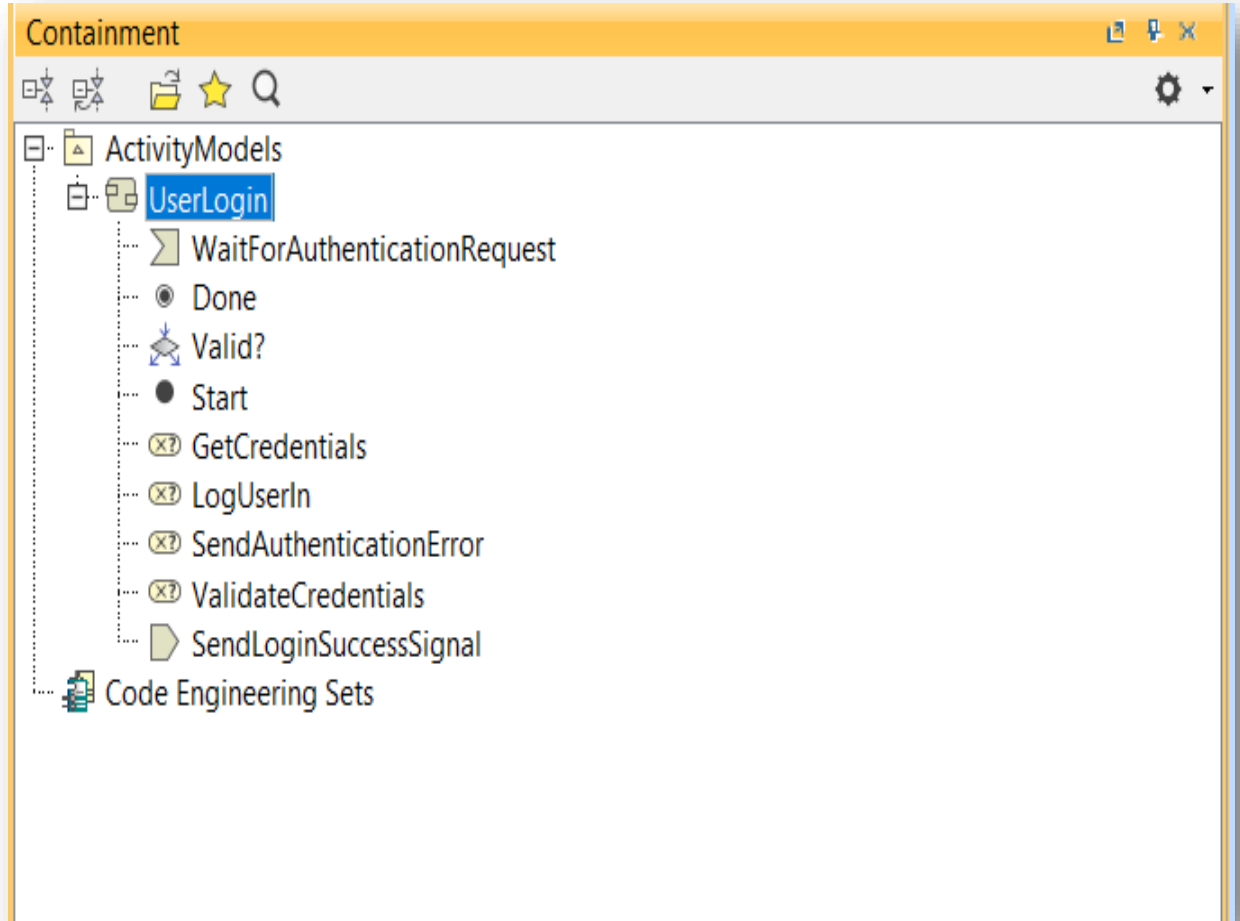
      <node xmi:type="uml:DecisionNode" name="Valid?"/>

      <node xmi:type="uml:OpaqueAction" name="LogUserIn"/>

    </packagedElement>

  </uml:Model>

</xmi:XMI>
```



Ways to Use Generative AI: Explainability



Explainability: Explain what the Requirements, Use Cases, and Activity Diagrams are Doing in 3 Bullet Form from the Information Above – “GPT-4”

Here's a summary of the requirements, use cases, and activity diagrams based on the information provided:

1. Requirements:

- The requirements define the system's behavior or features, such as how a user should be able to log in, what happens when the login fails multiple times, and so on. These requirements form the basis for the system's functionality and guide the development process.

2. Use Cases:

- The Use Cases (UC1 to UC5) outline various scenarios that occur within the system, each involving an actor (either a user or the system itself), preconditions, a flow of actions, and postconditions. They include user authentication, account locking after multiple failed login attempts, password reset, password encryption, and logging of login attempts.

3. Activity Diagrams:

- The activity diagrams (as represented in your XMI code) visually represent the workflow of a system or process, showing the sequence of activities or steps from start to end. They help in understanding the flow of control in the system. For example, the "UserLogin" activity diagram outlines the steps involved in a user logging into the system, including entering credentials, validating them, and either logging in or displaying an error, depending on validation success.

Explainability: VehicleRequirementDerivation.sysml

```
package VehicleRequirementDerivation {  
    import RequirementDerivation::*;  
  
    part vehicle {...
```

[View other drafts](#) 



Your package `VehicleRequirementDerivation` is a good example of how to use the Requirement Derivation language to derive vehicle requirements.

The package defines a `VehicleRequirementDerivation` part, which contains a `vehicle` part with attributes for the vehicle's mass, chassis mass, and engine mass. The package also defines a `MassRequirement` requirement, which has a `subject` attribute that identifies the mass to be constrained and a `massLimit` attribute that specifies the maximum allowable mass.

The package then defines three derived requirements: `vehicleMassRequirement`, `chassisMassRequirement`, and `engineMassRequirement`. Each of these requirements is a specialization of the `MassRequirement` requirement, and each one specifies a different subject mass.

The `derivation connection` section of the package connects the derived requirements to the original requirement. This section specifies that the `vehicleMassRequirement` requirement is the original requirement, and that the `chassisMassRequirement` and `engineMassRequirement` requirements are derived from it.

This package provides a good example of how to use the Requirement Derivation language to derive vehicle requirements. The language allows you to explicitly define the relationships between requirements, which can help to ensure that the requirements are complete and consistent.

Here is an example of how to use the `VehicleRequirementDerivation` package to derive vehicle requirements:

```
VehicleRequirementDerivation::VehicleRequirementDerivation vehicleRequirementDerivation;  
  
vehicleRequirementDerivation.vehicle.mass := 1000 kg;  
vehicleRequirementDerivation.vehicle.chassis.mass := 500 kg;  
vehicleRequirementDerivation.vehicle.engine.mass := 250 kg;  
  
vehicleRequirementDerivation.deriveRequirements();  
  
// The vehicleMassRequirement requirement will now be equal to 1000 kg.  
// The chassisMassRequirement requirement will now be equal to 500 kg.  
// The engineMassRequirement requirement will now be equal to 250 kg.
```

The `VehicleRequirementDerivation` package can be used to derive a wide variety of vehicle requirements. For example, you could use the package to derive requirements for the vehicle's performance, safety, and reliability.

MBSE Generative AI Takeaways

- AI techniques like reasoning facilitate model consistency, identification of issues, and automatic improvement recommendations
- Generative capabilities help rapidly create multiple model variants for trade-off analysis and design space exploration
- Models may be generated from requirements and code may be generated from models
- AI-assisted modeling and simulation can analyze system behavior and identify integration issues early
- **Automation frees up Engineers to focus on creative design rather than repetitive modeling tasks**



MBSE Generative AI Future Work

- Continue fine tuning custom open source LLMs for requirements verification and validation
- Explore training open source LLMs for SysML Version 2 generation
- Explore training and generation of LLMs to support more detailed activity diagram generation



Questions



References

- [What Is Generative AI? Everything You Need to Know \(webcluesinfotech.com\)](https://webcluesinfotech.com)
- <https://www.leewayhertz.com/generative-ai-use-cases-and-applications/>
- <https://alltogether.swe.org/2023/07/generative-ai-engineers/>
- [https://sebokwiki.org/wiki/Artificial Intelligence](https://sebokwiki.org/wiki/Artificial_Intelligence)
- <https://chat.openai.com/>
- <https://bard.google.com/>
- <https://ai.meta.com/llama/>
- A Survey of Large Language Models, <https://arxiv.org/pdf/2303.18223.pdf>
- A Comprehensive Survey of AI-Generated Content (AIGC): A History of Generative AI from GAN to ChatGPT, <https://arxiv.org/pdf/2303.04226.pdf>
- WHAT IS ARTIFICIAL INTELLIGENCE? <https://www-formal.stanford.edu/jmc/whatisai.pdf>
- <https://www.ibm.com/topics/neural-networks>
- <https://www.ibm.com/topics/machine-learning>
- <https://www.ibm.com/topics/artificial-intelligence>
- <https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html>
- <https://ai.meta.com/llama/>
- <https://arxiv.org/pdf/2304.13712.pdf><https://magazine.sebastianraschka.com/p/understanding-large-language-models>
- <https://cobusgreyling.medium.com/the-foundation-large-language-model-llm-tooling-landscape-8a849ebc7228>
- <https://base10.vc/post/generative-ai-mission-critical/>
- <https://www.assemblyai.com/blog/the-full-story-of-large-language-models-and-rlhf/>

