Future Consideration of Generative Artificial Intelligence (AI) for Systems Engineering Design

Ariel Mordoch
Systems Engineer, Enterprise MBSE
Defense, Space & Security (BDS)

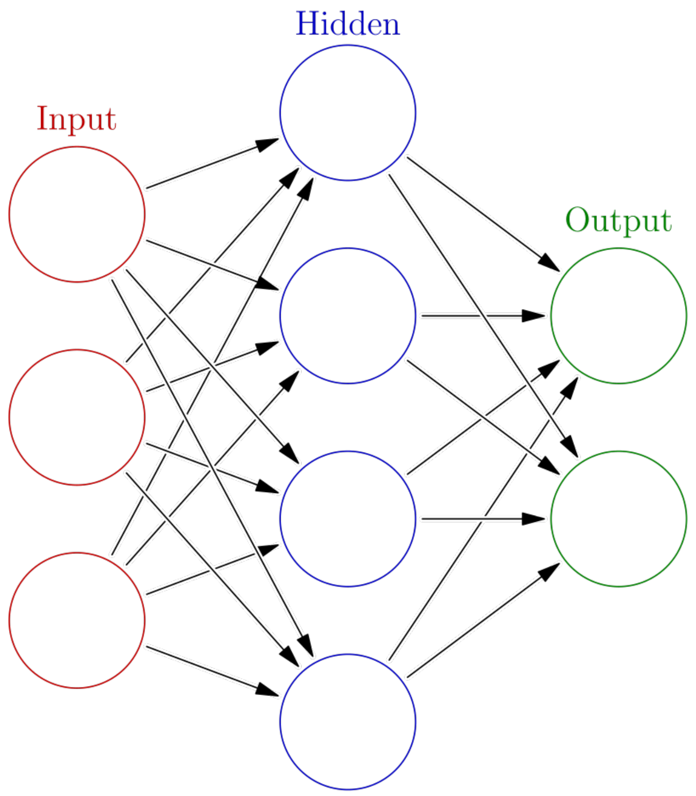
Andrew J. Gabel
Lead Systems Architect, Air Dominance
Defense, Space & Security (BDS)

Updated: 13 September 2023

Agenda

Machine Learning

- **Machine learning (ML) is a catch-all term for the development of algorithms by computers (rather than humans)**
- **Training is the process by which computers 'learn' and optimize these algorithms**
  - Genetic algorithm
  - Reinforcement learning
  - Supervised learning
  - Etc.
- **Models are the final algorithms produced by training**
- **The most popular models today are artificial neural networks**
  - Neural networks loosely mimic the human brain
  - They consist of "nodes" that transform inputs to outputs



https://en.wikipedia.org/wiki/Machine_learning#/media/File:Colored_neural_network.svg

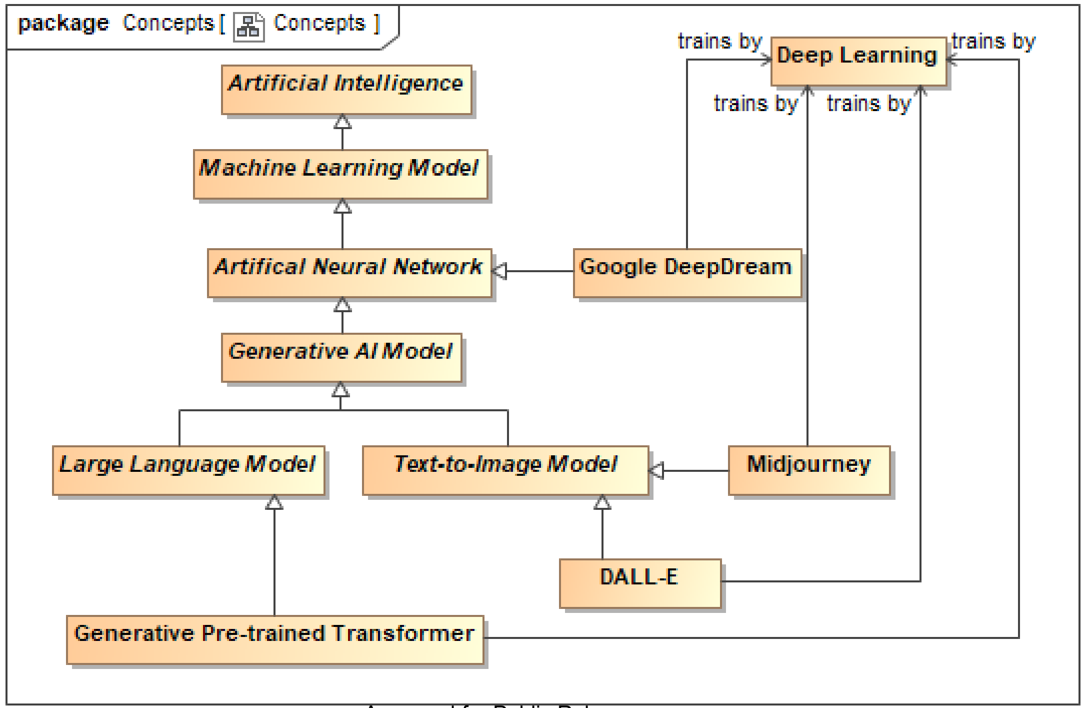
History of Machine Learning

- **Machine learning has actually been around for a while**
  - A machine learning algorithm for checkers was created at IBM in 1959[1]
- **With the advent of deep learning in the 2010s, ML became significantly more widespread[2]**
- **ML models have been used for years in areas such as**
  - Finance
  - Content recommendation
    - Facebook, Twitter, YouTube, etc.
  - Search
    - Google PageRank
  - Computer vision
    - Object identification
  - And more…

1. A. L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers," in *IBM Journal of Research and Development*, vol. 3, no. 3, pp. 210-229, July 1959, doi: 10.1147/rd.33.0210.
2. Graesser, Laura, and Wah Loon Keng. "Epilogue A. Deep Reinforcement Learning Timeline." *Foundations of Deep Reinforcement Learning: Theory and Practice in Python*. Addison-Wesley Professional, 2019.

Introduction – What is GenAI?

- Generative AI (or Gen AI) is a class of AI models built to generate text, images, etc.
- These models are trained on gargantuan datasets using machine learning
- Is Gen AI that new?
  - GPT was initially released in 2018
    - *ChatGPT uses GPT 3.5+*
- The recent explosion in AI interest began when ChatGPT became widely available

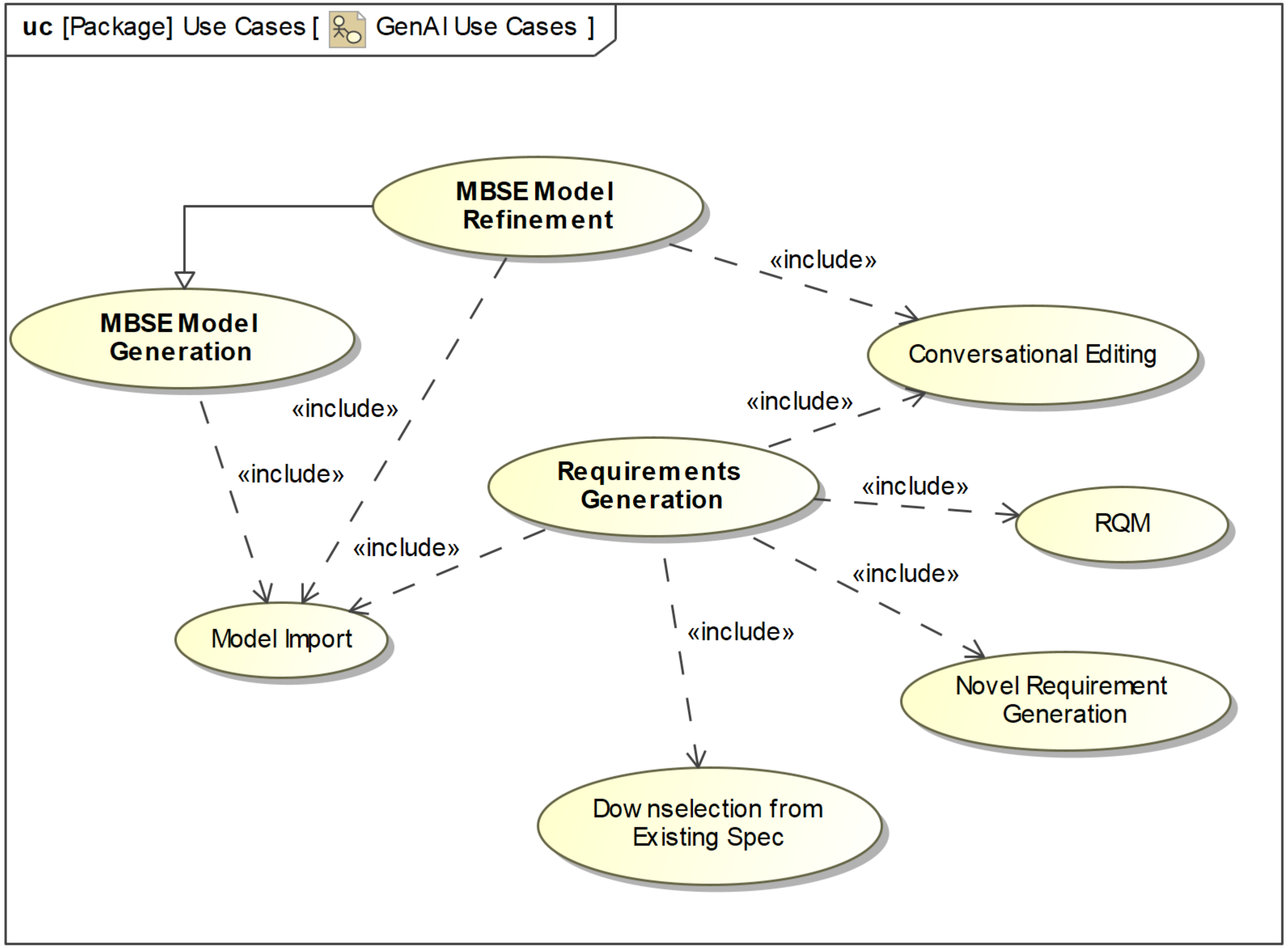


Large Language Models (LLMs)

LLMs predict text

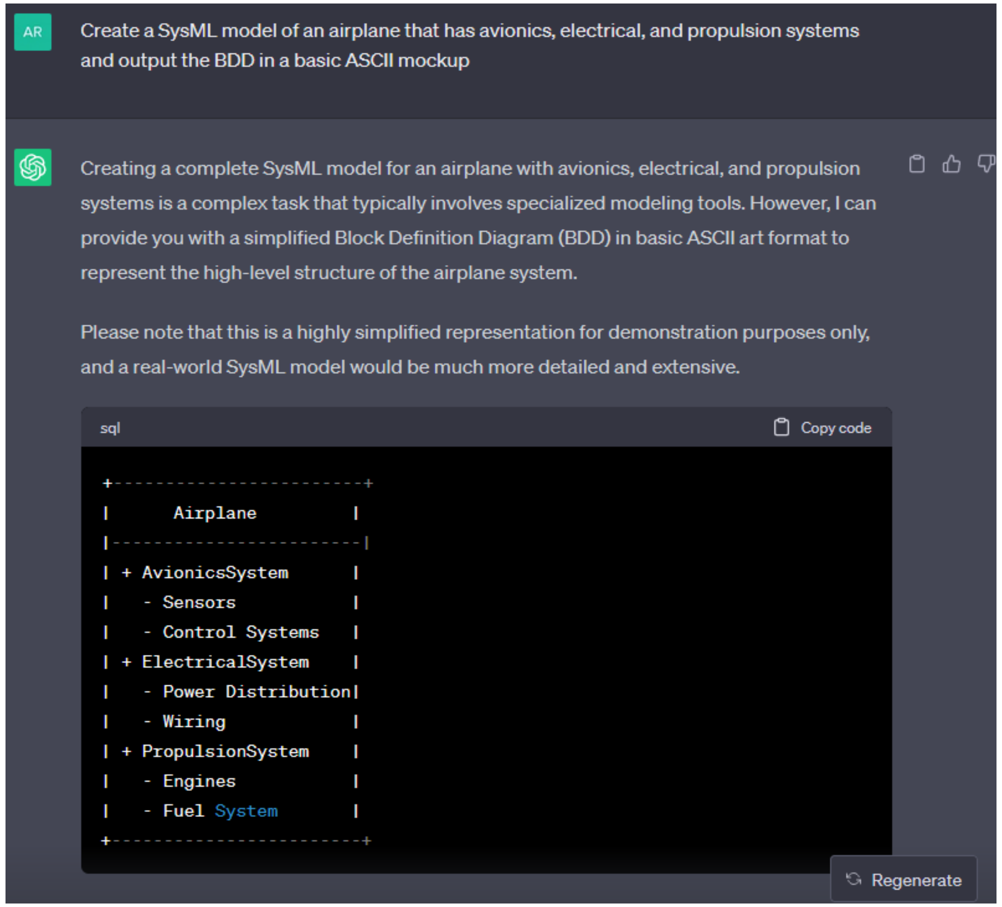
- LLMs are models that predict the next word based on your input
- LLMs are excellent for generating vast quantities of text
- GPT, the most well known LLM, can:
  - Hold conversations
  - Write essays
  - Pass the bar exam
  - Write code in virtually any programming language
  - Translate entire documents
- Virtually every large tech company now has an LLM (or is investing in one)

Use Cases for Gen AI in Systems Engineering



ChatGPT Model Generation

- ChatGPT "knows" about popular modeling languages
  - SysML (v1 and v2
  - UML
  - UAF
  - DoDAF
- And so it can generate models!


AR

Create a SysML model of an airplane that has avionics, electrical, and propulsion systems and output the BDD in a basic ASCII mockup

Creating a complete SysML model for an airplane with avionics, electrical, and propulsion systems is a complex task that typically involves specialized modeling tools. However, I can provide you with a simplified Block Definition Diagram (BDD) in basic ASCII art format to represent the high-level structure of the airplane system.

Please note that this is a highly simplified representation for demonstration purposes only, and a real-world SysML model would be much more detailed and extensive.

```
+-------------------------+
|      Airplane           |
|-------------------------|
| + AvionicsSystem        |
|   - Sensors             |
|   - Control Systems     |
| + ElectricalSystem      |
|   - Power Distribution|
|   - Wiring              |
| + PropulsionSystem      |
|   - Engines             |
|   - Fuel System         |
+-------------------------+
```

Model Generation Use Cases

- **Clean slate (new) model**
  - Rapid creation of complex model (airplane, car, etc.) with many parts, interfaces, etc. in seconds
  - Jumping-off point for further modeling
- **Model Refinement**
  - Modify an existing model quickly with a back-and-forth conversation
  - For example:
    - Modifying interfaces based on a textual description from an ICD
    - Adding parts and value properties to a block
- **Conversion of document-based descriptions to structural and behavioral definitions**
  - Many modeling projects involve some legacy encoded in documents
  - ChatGPT can readily read in behavioral or structural descriptions and turn them into models
- **Overall theme: <u>massive potential to improve productivity</u>**

Concerns with the Use of AI

- **Information protection (IP, CUI, classified, etc.)**
  - Storage, transmission, etc. requirements apply
  - May not be met by commercially available LLMs
  - Use of AI may not be allowed for this reason (policy dependent)
- **Shortcutting (clean-slate use case)**
  - In early modeling phases, much time is spent learning the system
  - AI enables skipping this learning
  - Model should be used as a starting point, not as the end goal
- **Model Quality**
  - Generally, GPT is only good as a starting point
  - Generated models will certainly need to be modified and improved
  - Easy to rapidly create many low quality models

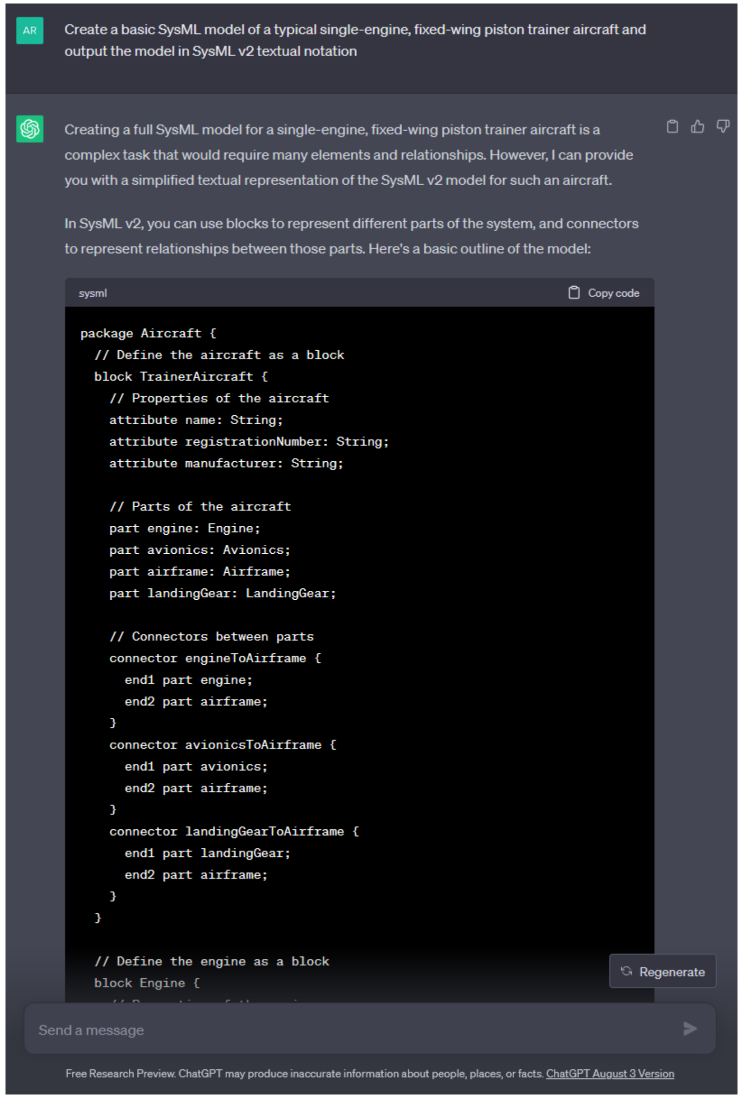
Ingesting Generated Models

- ChatGPT can provide textual descriptions, ASCII art, etc. all day
- **How can we get it to produce models that we can ingest into a modeling tool?**
  - OMG XMI
    - XMI is an XML schema maintained by OMG for model interchange
    - Most modeling tools can ingest models stored in XMI format
    - Not human readable, but easy to parse
  - SysML v2 Textual Notation
    - Human readable, straightforward
    - No widely available parsers
    - SysML 2 not yet supported by engineering tools

GPT Output Formatting

- GPT can readily output structured text
  - MBSE models are no different
- Both methods can work depending on what you want to implement
- **GPT often gets very, very close, but has minor errors**
  - Can prove detrimental in highly structured formats (XML)
- **GPT requires the right system messages (instructions/directives) to fully get things right**

GPT Output Formatting cont'd.



AR Create a basic SysML model of a typical single-engine, fixed-wing piston trainer aircraft and output the model in SysML v2 textual notation

Creating a full SysML model for a single-engine, fixed-wing piston trainer aircraft is a complex task that would require many elements and relationships. However, I can provide you with a simplified textual representation of the SysML v2 model for such an aircraft.

In SysML v2, you can use blocks to represent different parts of the system, and connectors to represent relationships between those parts. Here's a basic outline of the model:

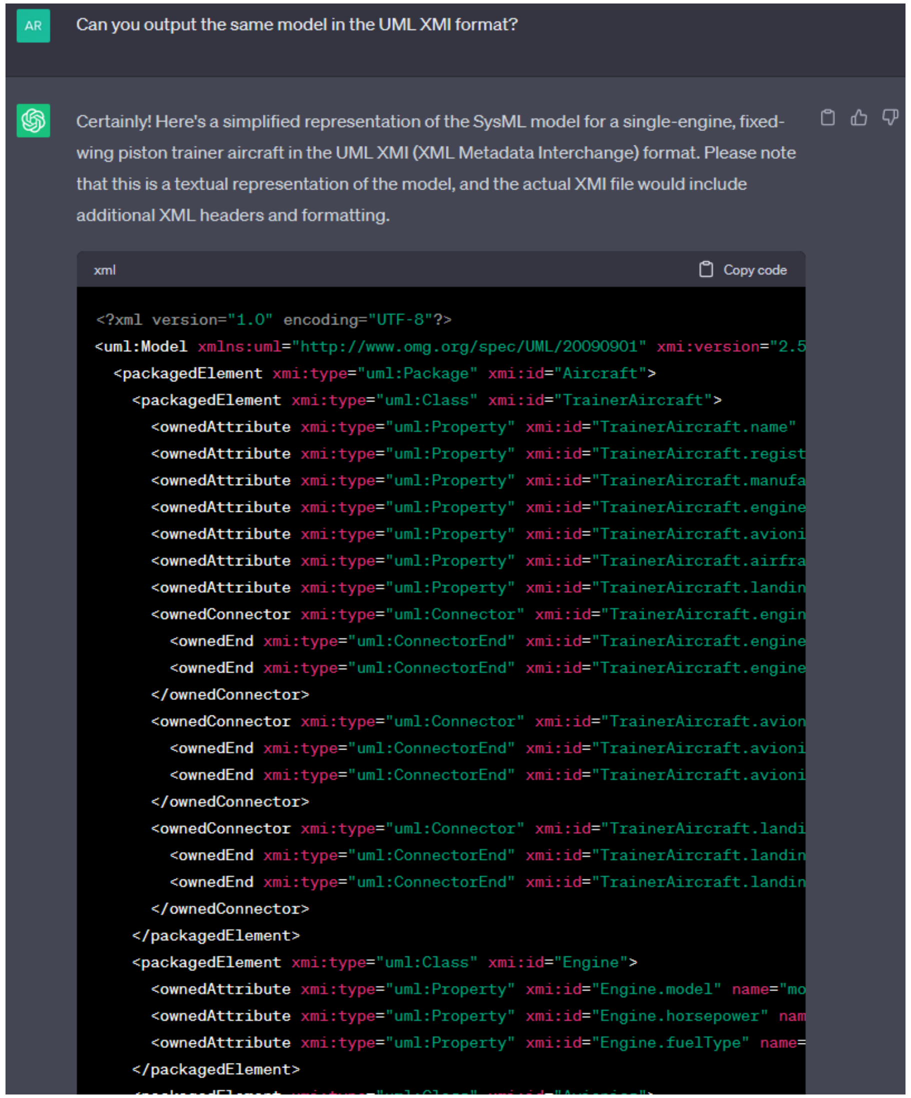
sysml | Copy code

```
package Aircraft {
  // Define the aircraft as a block
  block TrainerAircraft {
    // Properties of the aircraft
    attribute name: String;
    attribute registrationNumber: String;
    attribute manufacturer: String;

    // Parts of the aircraft
    part engine: Engine;
    part avionics: Avionics;
    part airframe: Airframe;
    part landingGear: LandingGear;

    // Connectors between parts
    connector engineToAirframe {
      end1 part engine;
      end2 part airframe;
    }
    connector avionicsToAirframe {
      end1 part avionics;
      end2 part airframe;
    }
    connector landingGearToAirframe {
      end1 part landingGear;
      end2 part airframe;
    }
  }

  // Define the engine as a block
  block Engine {
```

Regenerate

Send a message

Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT August 3 Version



AR Can you output the same model in the UML XMI format?

Certainly! Here's a simplified representation of the SysML model for a single-engine, fixed-wing piston trainer aircraft in the UML XMI (XML Metadata Interchange) format. Please note that this is a textual representation of the model, and the actual XMI file would include additional XML headers and formatting.

xml | Copy code

```
<?xml version="1.0" encoding="UTF-8"?>
<uml:Model xmlns:uml="http://www.omg.org/spec/UML/20090901" xmi:version="2.5
  <packagedElement xmi:type="uml:Package" xmi:id="Aircraft">
    <packagedElement xmi:type="uml:Class" xmi:id="TrainerAircraft">
      <ownedAttribute xmi:type="uml:Property" xmi:id="TrainerAircraft.name"
      <ownedAttribute xmi:type="uml:Property" xmi:id="TrainerAircraft.regist
      <ownedAttribute xmi:type="uml:Property" xmi:id="TrainerAircraft.manufa
      <ownedAttribute xmi:type="uml:Property" xmi:id="TrainerAircraft.engine
      <ownedAttribute xmi:type="uml:Property" xmi:id="TrainerAircraft.avioni
      <ownedAttribute xmi:type="uml:Property" xmi:id="TrainerAircraft.airfra
      <ownedAttribute xmi:type="uml:Property" xmi:id="TrainerAircraft.landin
      <ownedConnector xmi:type="uml:Connector" xmi:id="TrainerAircraft.engin
        <ownedEnd xmi:type="uml:ConnectorEnd" xmi:id="TrainerAircraft.engine
        <ownedEnd xmi:type="uml:ConnectorEnd" xmi:id="TrainerAircraft.engine
      </ownedConnector>
      <ownedConnector xmi:type="uml:Connector" xmi:id="TrainerAircraft.avion
        <ownedEnd xmi:type="uml:ConnectorEnd" xmi:id="TrainerAircraft.avioni
        <ownedEnd xmi:type="uml:ConnectorEnd" xmi:id="TrainerAircraft.avioni
      </ownedConnector>
      <ownedConnector xmi:type="uml:Connector" xmi:id="TrainerAircraft.landi
        <ownedEnd xmi:type="uml:ConnectorEnd" xmi:id="TrainerAircraft.landin
        <ownedEnd xmi:type="uml:ConnectorEnd" xmi:id="TrainerAircraft.landin
      </ownedConnector>
    </packagedElement>
    <packagedElement xmi:type="uml:Class" xmi:id="Engine">
      <ownedAttribute xmi:type="uml:Property" xmi:id="Engine.model" name="mo
      <ownedAttribute xmi:type="uml:Property" xmi:id="Engine.horsepower" nam
      <ownedAttribute xmi:type="uml:Property" xmi:id="Engine.fuelType" name=
    </packagedElement>
```

https://chat.openai.com/share/c1562053-889d-42d7-b225-4667f8b670b4

Fine Tuning

- **Some LLMs can be further trained in a process called *fine tuning***
  - Decreased prompt complexity
  - Better reliability in producing desired responses
  - Outputting a particular style or format
- **Ex.: Training GPT on high-quality MBSE models**
  - To learn your organization’s modeling patterns and style
  - So it then outputs models in your organization’s style

LLM Requirements Generation

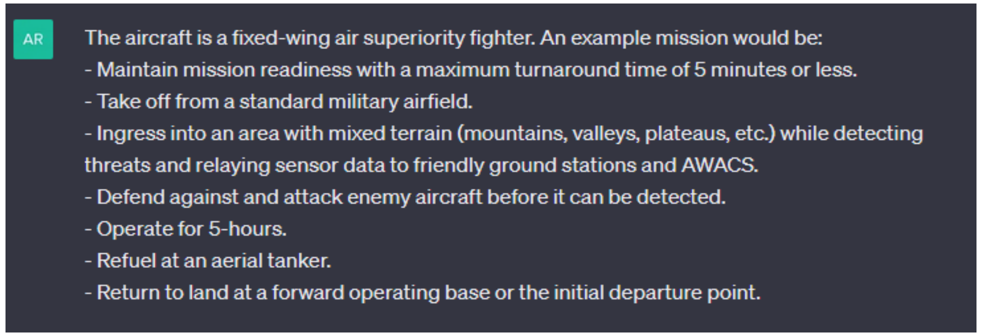
- **Text-based requirements are obviously easy targets for LLMs**
- However, quality is a big concern
- **GPT in particular needs to be coached before it produces high quality requirements**
- Coaching can come from
  - Providing existing requirements documents
  - Putting in examples of high-quality requirements
  - Providing a requirements writing work instruction/process

Requirements Generation Use Cases

- **Initial slate of requirements**
  - From textual descriptions (CONOPS, DRMs, etc.), creation of an initial set of requirements
- **Refinement of existing requirements/RQM**
  - Provide an existing set of requirements
  - Ask for low quality requirements
  - Propose text changes
  - Identify conflicting requirements
- **Selecting relevant requirements for a new project**
  - Often start from previous requirements instead of a clean slate
  - LLMs could help down select, trim down, and edit requirements
    - Given a description of the old system vs. the new one

Requirement Example

- We provided ChatGPT with the following example*:
  - Air superiority fighter
  - Capable of landing at short and standard length airfields
  - Operation in a mountainous environment
  - 5 hour mission time, 5 minute turnaround time
  - Aerial refuel
  - Some level of stealth capability


AR

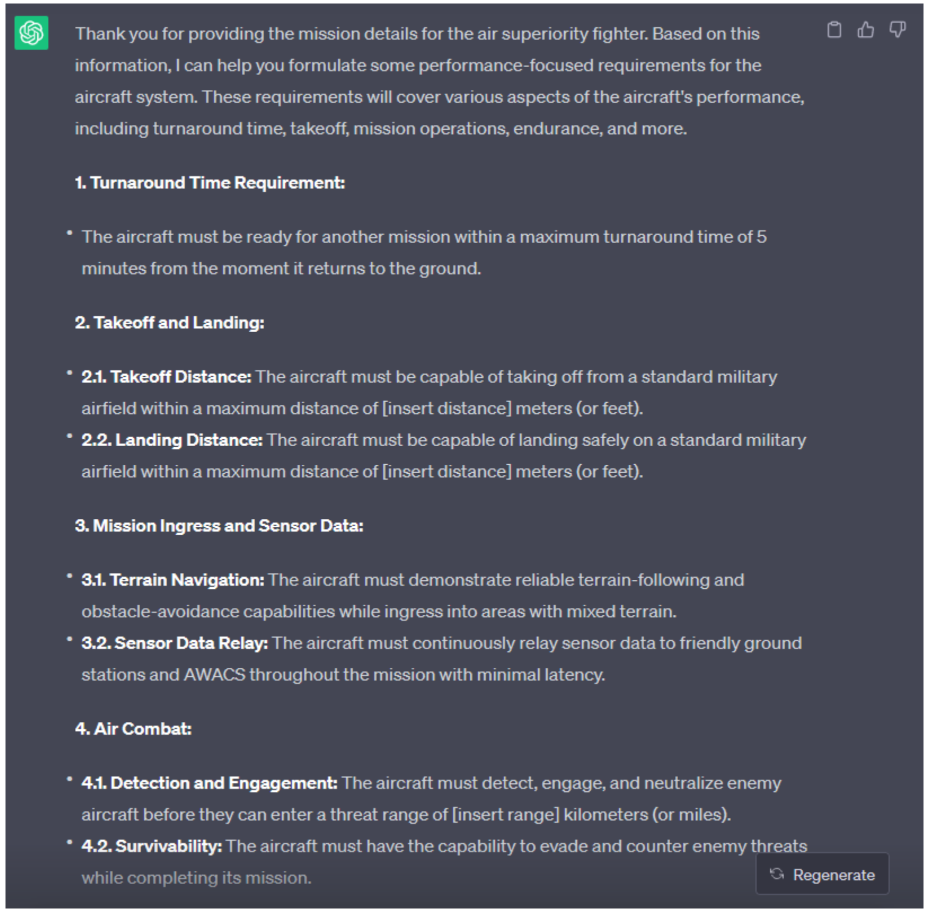
The aircraft is a fixed-wing air superiority fighter. An example mission would be:
- Maintain mission readiness with a maximum turnaround time of 5 minutes or less.
- Take off from a standard military airfield.
- Ingress into an area with mixed terrain (mountains, valleys, plateaus, etc.) while detecting threats and relaying sensor data to friendly ground stations and AWACS.
- Defend against and attack enemy aircraft before it can be detected.
- Operate for 5-hours.
- Refuel at an aerial tanker.
- Return to land at a forward operating base or the initial departure point.

https://chat.openai.com/share/f8a67b97-7a3e-4aab-b7b6-518d1f46d0e1

**Does not represent a real aircraft; example only.*

Requirement Example cont'd

- Initial response used "must" statements
- But had all the requested requirement topics
  - As well as reliability, which wasn't mentioned



Thank you for providing the mission details for the air superiority fighter. Based on this information, I can help you formulate some performance-focused requirements for the aircraft system. These requirements will cover various aspects of the aircraft's performance, including turnaround time, takeoff, mission operations, endurance, and more.

1. Turnaround Time Requirement:

- The aircraft must be ready for another mission within a maximum turnaround time of 5 minutes from the moment it returns to the ground.

2. Takeoff and Landing:

- **2.1. Takeoff Distance:** The aircraft must be capable of taking off from a standard military airfield within a maximum distance of [insert distance] meters (or feet).
- **2.2. Landing Distance:** The aircraft must be capable of landing safely on a standard military airfield within a maximum distance of [insert distance] meters (or feet).

3. Mission Ingress and Sensor Data:

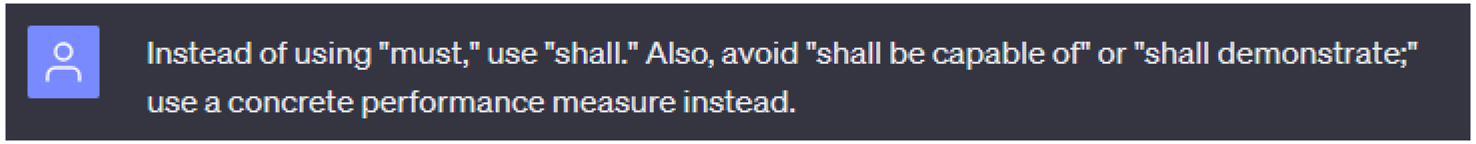
- **3.1. Terrain Navigation:** The aircraft must demonstrate reliable terrain-following and obstacle-avoidance capabilities while ingress into areas with mixed terrain.
- **3.2. Sensor Data Relay:** The aircraft must continuously relay sensor data to friendly ground stations and AWACS throughout the mission with minimal latency.

4. Air Combat:

- **4.1. Detection and Engagement:** The aircraft must detect, engage, and neutralize enemy aircraft before they can enter a threat range of [insert range] kilometers (or miles).
- **4.2. Survivability:** The aircraft must have the capability to evade and counter enemy threats while completing its mission.

Requirement Example cont'd.

- The aircraft must…
  - …demonstrate reliable terrain-following and obstacle-avoidance capabilities while ingress into areas with mixed terrain.
  - …demonstrate a reliability rate of at least [insert percentage], ensuring that it completes missions without critical system failures.
  - …have the capability to evade and counter enemy threats while completing its mission.
- **Not particularly high quality**
- Further prompt to refine:


Instead of using "must," use "shall." Also, avoid "shall be capable of" or "shall demonstrate;" use a concrete performance measure instead.

Requirements Example cont’d.

- **After just one refinement, the aircraft shall**
  - …maintain terrain-following and obstacle-avoidance capabilities during ingress into areas with mixed terrain with a minimum clearance of 100 meters from obstacles.
  - …achieve a reliability rate of at least 95%, ensuring that it completes missions without critical system failures.
  - …shall have the capability to evade and counter enemy threats while maintaining mission integrity.
- Still imperfect
  - But quicker than typing
- Example chat
  - https://chat.openai.com/share/f8a67b97-7a3e-4aab-b7b6-518d1f46d0e1

Requirement Generation Considerations

- **Quality**
  - As we've seen, coaxing is required
- **Hallucination**
  - GPT does not "know" anything
  - As you increase specificity/domain knowledge, the likelihood of hallucination increases
- **Acceptability**
  - Will the program accept AI generated requirements?
  - Will your customer accept AI generated requirements?
- **Accountability**
  - Typically have thousands of requirements
  - But they (hopefully) all had a purpose
  - Will designers comply with generated requirements they didn't come up with or have a hand in creating?

Integration with SE Tools

- **Using the same ideas from earlier:**
  - GPT can export requirements as an MBSE model
- **ReqIF**
  - Another OMG standard GPT can readily convert to
  - Most tools have native import/export
- **Tabular/CSV**
  - Classic spreadsheet
  - Importable by pretty much anything

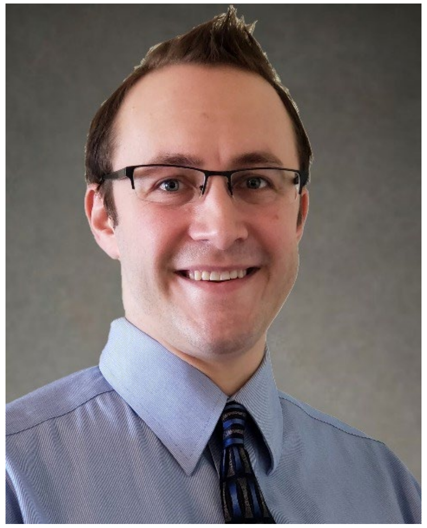
Conclusions

- **LLMs can significantly enhance systems engineers' productivity**
  - MBSE model generation and refinement
  - Requirements generation
- **Engineering judgement is still required**
  - Hallucination
  - LLMs don't "know" things
- **Context is everything**
  - The more context given, the better the results
- **Security and information protection concerns remain a barrier**

About the Authors

Ariel Mordoch

Ariel joined Boeing and the Enterprise MBSE Capability team in July 2021. Since then, he has supported many MBSE efforts, including requirements management, integrated simulations with external tools, architecture development, integrating AI with Systems Engineering, and integrating specialty engineering disciplines into MBSE models. Ariel has also authored several Cameo/MSOSA plugins and garnered a reputation within Boeing as a Cameo/MSOSA expert. Ariel's background is in Aerospace Engineering, in which he completed his B.S. at the Georgia Institute of Technology in 2020.



Andrew J. Gabel

Andrew is an experienced Boeing engineer with over 11 years in the industry. His expertise lies in merging Software Engineering with Systems Engineering to gain a holistic grasp of the systems he works on. Currently, he leads efforts to consult on Model Based Systems Engineering (MBSE) with experience in over 10 programs. He earned a Master of Engineering Management from the University of Nebraska – Lincoln in 2023 and holds a Bachelor of Science in Computer Science and Engineering Physics from Kansas Wesleyan University, awarded in 2012.