Massachusetts Institute of Technology

Model-Based Systems Engineering: Documentation and Analysis

Week 3: Critiquing an MBSE Approach

Project

Name

Jane Doe

Massachusetts Institute of Technology

Week 3 Project

Overview

In Week 3, you will choose, download, and critique one of the projects we provided.
Please note that the project provided combines weeks 1 and 2 project submissions – good practices– from a previous course so you will not be conducting a critique of a full model.

Each document contains real cases from different industries, so you can select the one you are more interested in.

Project Objectives

- To practice creating quality critiques

You should presume that the rationale for your critique is to: "**Evaluate whether a project should adopt the MBSE approach or stay with the status-quo.**"

REQUIRED STEPS

Step 1: Choose one of the good practices provided

Step 2: Download and read the project

Step 3: Critique the project using the template from the document you downloaded

Step 4: Add your name and surname to the project, and save your file with your name and surname too, i.e., [**W3Critique_Name_Surname**]

Step 4: Evaluate the two critiques from your peers that the platform will automatically assign you

Step 1: Choose System and Define Scope

What is your system? What is in scope for the MBSE effort for your chosen system?

For this project I am going to try to apply the MBSE to build a software system for an automated teller machine (ATM). An ATM is a complex system that has a lot of different components. It has a card reader then reads the credit/debit cards of the user. It has a UI that interacts with the user and allows him/her to check account details and withdraw cash. It also communicates with the bank to get the user's information and to process his financial transactions. Apart from this, there are many other subcomponents to ensure security, transaction compliance, etc.

The purpose of this MBSE model is to help manage the development of this complex system. In particular, we want to manage the requirements of all the different subsystems, identify any conflicts, and manage the code development. Furthermore, the model should ensure that all operating conditions and boundary conditions are met by the software, including the financial regulations and compliance requirements. The MBSE strategy will maintain a single source of information, incorporate the basic validations, and facilitate information flow across different tools used by teams.

Step 2: Define MBSE Approach

How would the approach to this project change with MBSE?

One of the biggest challenges for the project is to allow information flow between different teams such as software development, hardware/firmware teams, UI designer, testing and product management. This will be facilitated by the MBSE approach. The MBSE model will connect with software tools used by different teams and facilitate information flow. Therefore, if the product teams make a change to the requirements in the product management tool, then those changes will be communicated instantly to the development team and the testing team by updating the software that those teams use for this project. This will ensure that everyone has the same, up to date information at any given time. If there are any conflicts, then teams can mutually decide and roll back the changes if required. To facilitate this flow, our MBSE approach needs to tackle three key challenges:

1. How information will be communicated from one team to another. As mentioned earlier, teams use different tools. The MBSE model needs to integrate with these tools and update them whenever there is a change. There are two parts to enable this. The first part is to monitor these tools to check if users are making any updates. If they have made updates and those updates satisfy the basic sanity check mentioned below, then this information needs to propagate to all the tools in the format that the different tools understand.
2. Once the MBSE model connects to the various software tools involved we also need to define rules that govern how changes to one software affect changes to others. Different tools don't speak the same language so there needs to be a translator that can relay the information. This job will be done by the MBSE model, and is the biggest part of the MBSE approach.
3. MBSE models need to have some basic sanity checks that would be required for any change so that it does not create downstream issues. An example validation could be to check if all the components of the system are performing at least one task after the change is made. Another validation could be to check if the change is contradicting other requirements or features.

Step 3: Define MBSE Purpose

What are the purposes of your MBSE effort? Describe the financial and non-financial benefits you expect.

An ATM deals with financial transactions, and therefore the tolerance for errors is pretty low. Organizations face two very big challenges.

1. They cannot be very flexible about updating requirements later in the development cycle since it might have adverse consequences. Therefore they have to be very rigid, and sometimes they miss out on releasing a lot of cool features that they could offer to customers.
2. They have to spend a lot of money testing the software because the tolerance for error in financial transactions is very low.

Both these challenges translate to lower profits, either in the form of increased costs or reduced revenue. The MBSE approach tries to address both challenges. It creates better visibility through proper information flow. Therefore, organizations can make better tradeoff decisions when dealing with change requests or requirement updates. In addition, that reduces testing costs significantly because MBSE performs validations and enables automation.

Step 4: List Major Tenets of MBSE

Describe how will you model the system. Briefly describe your approach to each of the major tenets of MBSE.

Central Model or Federation of Models:
We are trying to group the outputs of various software used by the different teams because each software performs a series of complex functions that could be treated as a model in its own right. All these models connect with the MBSE model, and people keep using the software tools that they are using today. Therefore, our strategy is a federation of models.

Model Views
The MBSE approach will accumulate the information and notify each stakeholder whenever there is a change made by someone else in the system. These could be in the form of alerts generated using email and in-software notifications. In addition, the MBSE strategy will set software triggers or hooks into the various software tools to update them whenever it finds stale information in any of them. This way users will have the most up to date information.

Model repository or library
As mentioned before, our MBSE approach will work as a translator for various software tools. This translation function could be built out in a modular way, so that we can add and remove software programs per will. The translation rules for each software program will be different and can be stored in a library or repository.

Standards and Patterns
The MBSE strategy will rely on web services and restful API standards to communicate with various software components. It would utilize web hooks to listen to any changes made by team members. It will also use triggers to update the information in the relevant software. These are example software standards used for communications.

Model Checking: Logically Verifiable Rules or Tests
As mentioned earlier our MBSE approach will have certain sanity checks so that people don't make changes willy-nilly.

Ontology
For our software development system there are certain definitions and terminologies that are used throughout the product development organization such as features, bugs, user stories, technical specification, and code bases. These entities have defined relationships. The MBSE strategy will use these existing ontologies in its approach.

MBSE Methodology
As defined above, the MBSE approach will have four parts: a listener that will find when an update is made to a software component using webhooks, a translator that translates the input from one software to a format understandable by others, a trigger to update the other software components with the changes, and a sanity check on the updates to make sure they are consistent with the overall system.

Step 5: Identify the Most Important Qualities of Great Models

Reflect on the qualities of great models – what are the top three you are concerned about for your MBSE approach?

Availability of Interfaces: One of the key success factors for the MBSE approach is its availability of interfaces. Since we need to talk to other software tools, each potentially having different interfaces, our MBSE strategy should support most software standards for communication, if not all that are used currently.

Internally Consistent: One of the biggest goals of our MBSE strategy is to facilitate information flow and to maintain the most up to date and current information across the project. To achieve this the model needs to be internally consistent. If multiple software components are making changes at the same time, it should identify what changes are being made by which software and make updates accordingly. No one should be able to update stale information in the model accidentally.

Verification & Validation With Models: Our MBSE strategy is heavily reliant on our model to perform sanity tests to ensure that we always have a complete and consistent system. This quality is another important enabler for other qualities like model fidelity and credibility.

Step 6: Identify Systems Engineering Tasks

Systems engineering has a variety of different tasks depending upon its role in the organization such as interface management, change management, and to facilitate information transfer. Which of these or other tasks in your view are applicable to your chosen MBSE strategy?

In software development we don't really use the term systems engineering explicitly, however the tasks mentioned above are generally performed by software engineers/developers. In bigger projects where multiple development teams are involved, some of the developers have to work on interface management to enable interaction between two disjoint software components. Another role that sometimes developers need to deal with is change management, since they are the ones building the system, they often find issues and glitches in initial specifications during development and have to take the lead communicating these issues and getting them fixed from the right stakeholders. Both these tasks would be better supported by our MBSE approach since it allows for better information flow in the system.

Step 1: Develop Five Queries for Your System

As you know by now, models provide a good deal more than just a set figures. Models are stored in repositories with a defined data structure. Like databases, the model repository makes it possible to query the model for specific information, e.g. an impact analysis when changing a requirement. SysML doesn't define a query language and most modeling tools allow to write a script to query the model. You can write queries like "are all actions allocated to parts?", "are all requirements satisfied?", and so on.

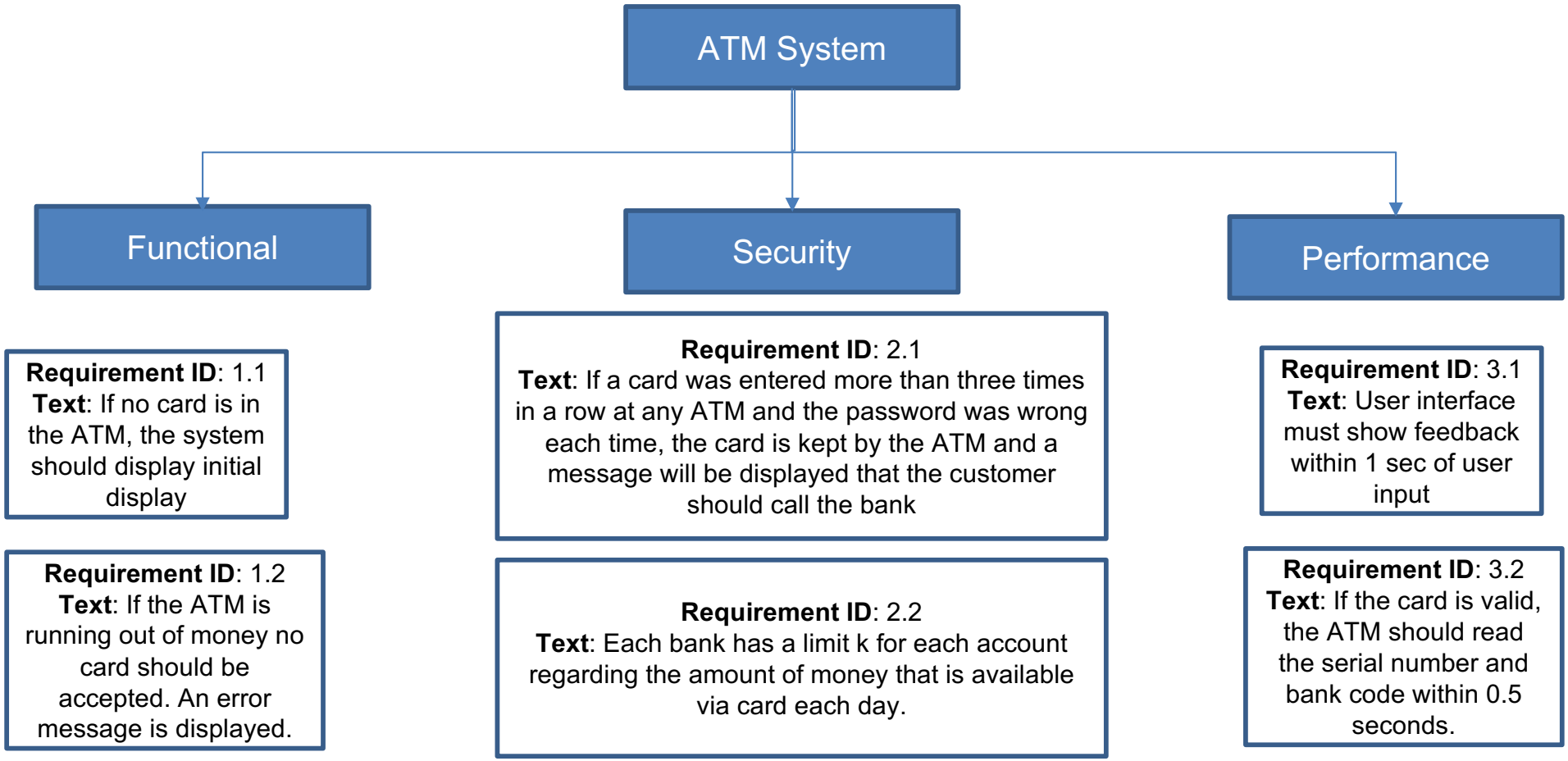
If you had a full data model available for your system, what would be five of the most important queries you would write to inform your system engineering functions?

Query	Rationale
Do I have requisite functions of the system assigned to components of the system?	*Making sure that all the functions are allocated*
What is the requirement traceability through the system?	*Tracing the requirements to the source and how they are applied through the system*
Which components are leveraged in the value delivery path of a particular function?	*Checking the components which are involved in executing a particular function*
How many instances of a component are used in the overall system?	*Checking the total number of particular components needed for one system (X washers in whole assembly)*
What is the total number of unique components/modules in the overall system?	*Checking the total component list*

Massachusetts Institute of Technology

Step 2: Develop the Requirement Diagram

Here, we are going to provide examples for an ATM system.

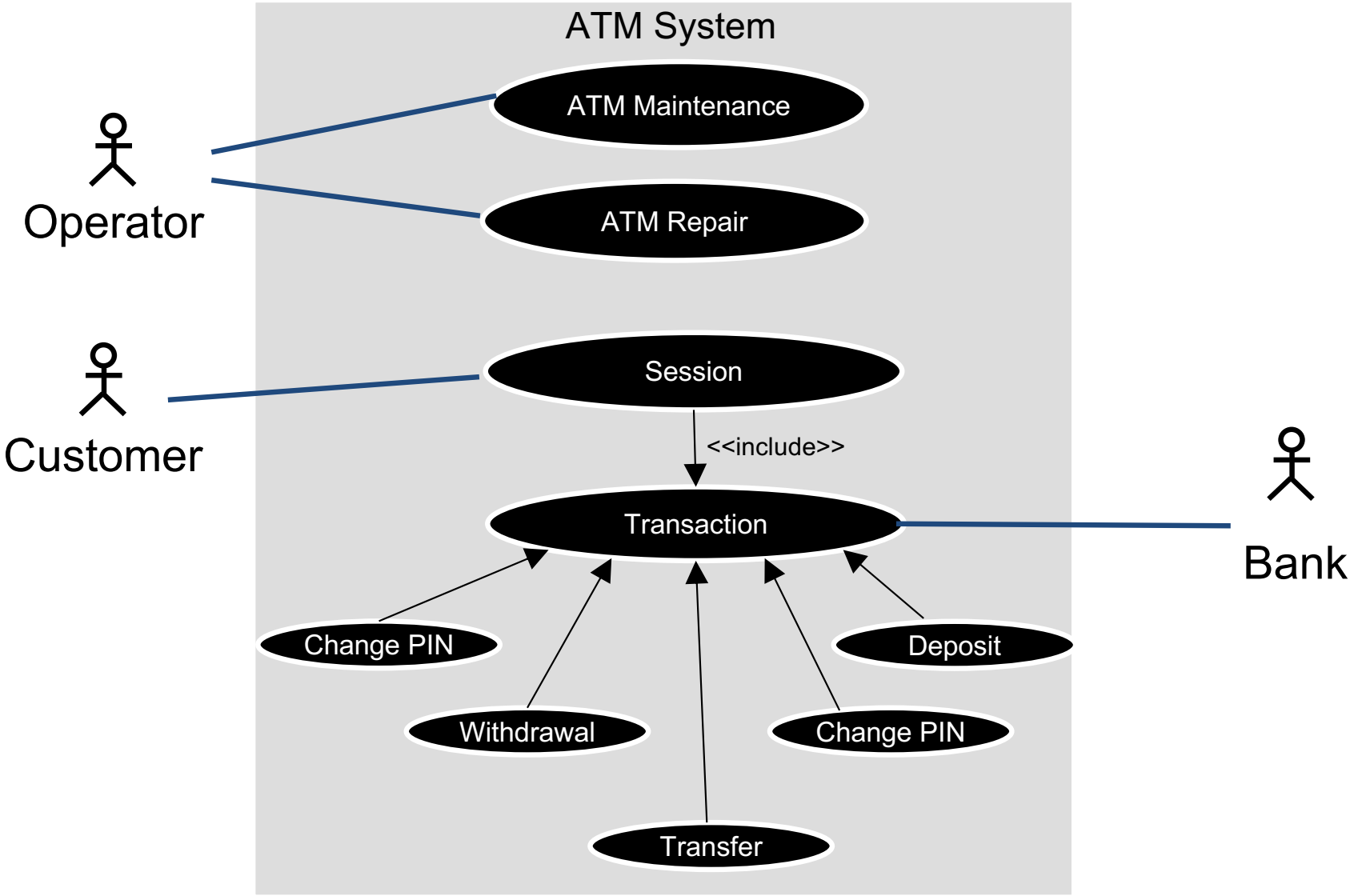


Note: This is just a sample diagram done without using a commercial SysML software. We just want you to get key ideas without getting too deep in syntax.

Massachusetts Institute of Technology

Step 3: Develop the Use Case Diagram

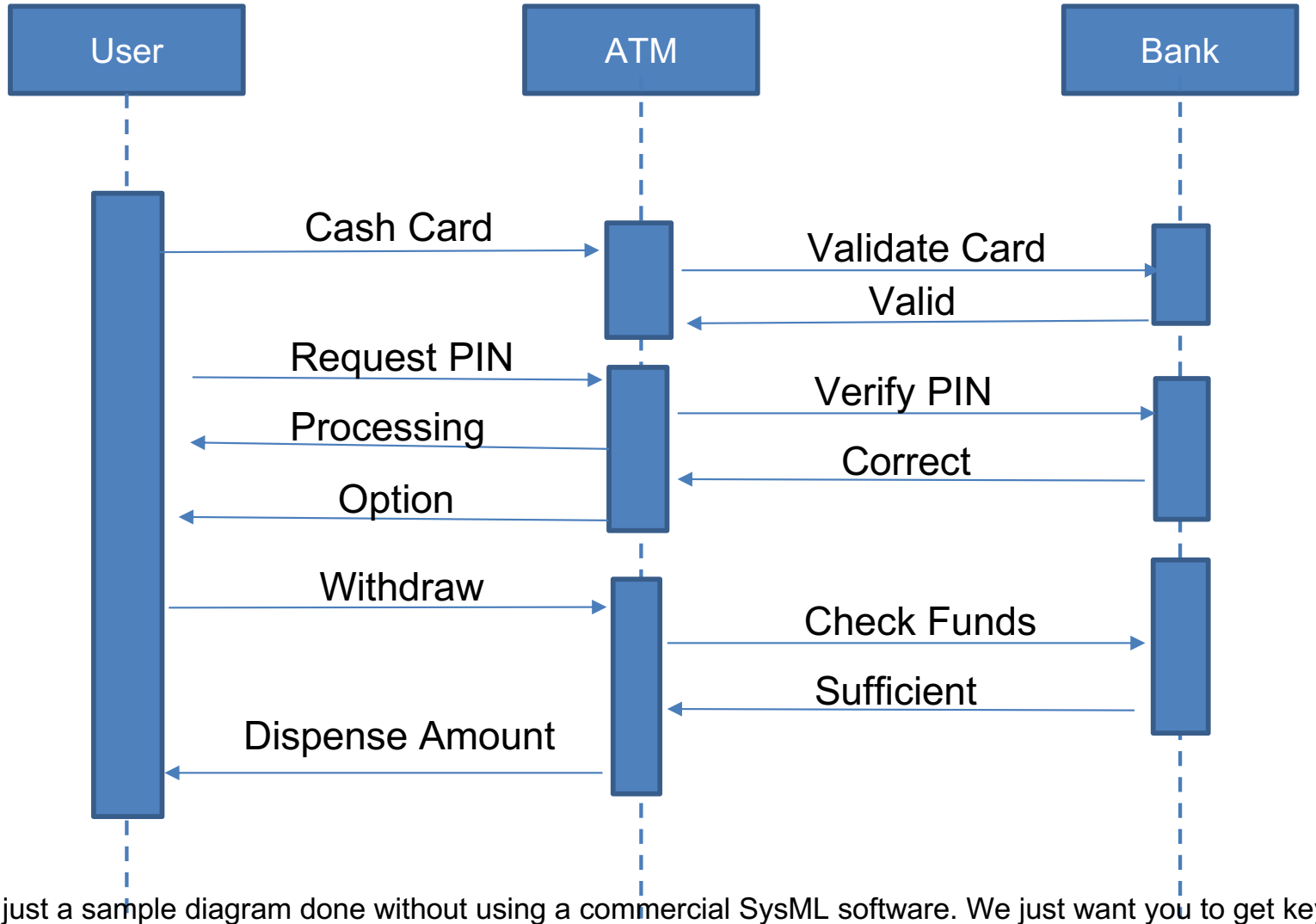
For the system you chose in Week 1, please create a use case diagram. Please feel free to leverage the format below or create your own.



Note: This is just a sample diagram done without using a commercial SysML software. We just want you to get key ideas without getting too deep in syntax.

Step 4: Develop the Behavior ~~or Structure~~ Diagram

Here, we provide a sample sequence diagram for an ATM system (Note: There can be many sequence diagrams based on different functions in consideration)

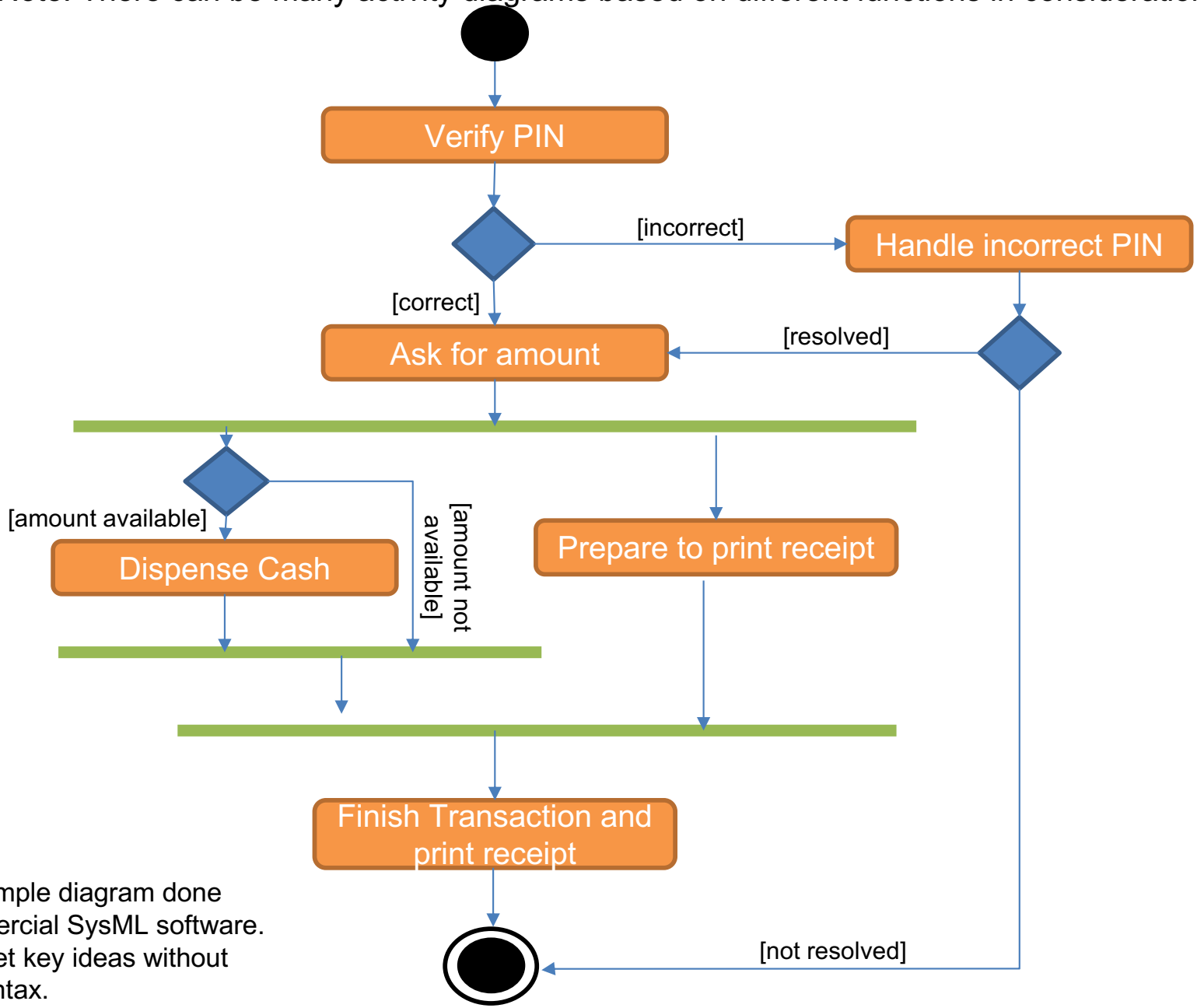


Note: This is just a sample diagram done without using a commercial SysML software. We just want you to get key ideas without getting too deep in syntax.

Massachusetts Institute of Technology

Step 4: Develop the Behavior ~~or Structure~~ Diagram

Here, we provide another sample (you need to do only one) of a behavior diagram for an ATM system – Activity Diagram (Note: There can be many activity diagrams based on different functions in consideration)

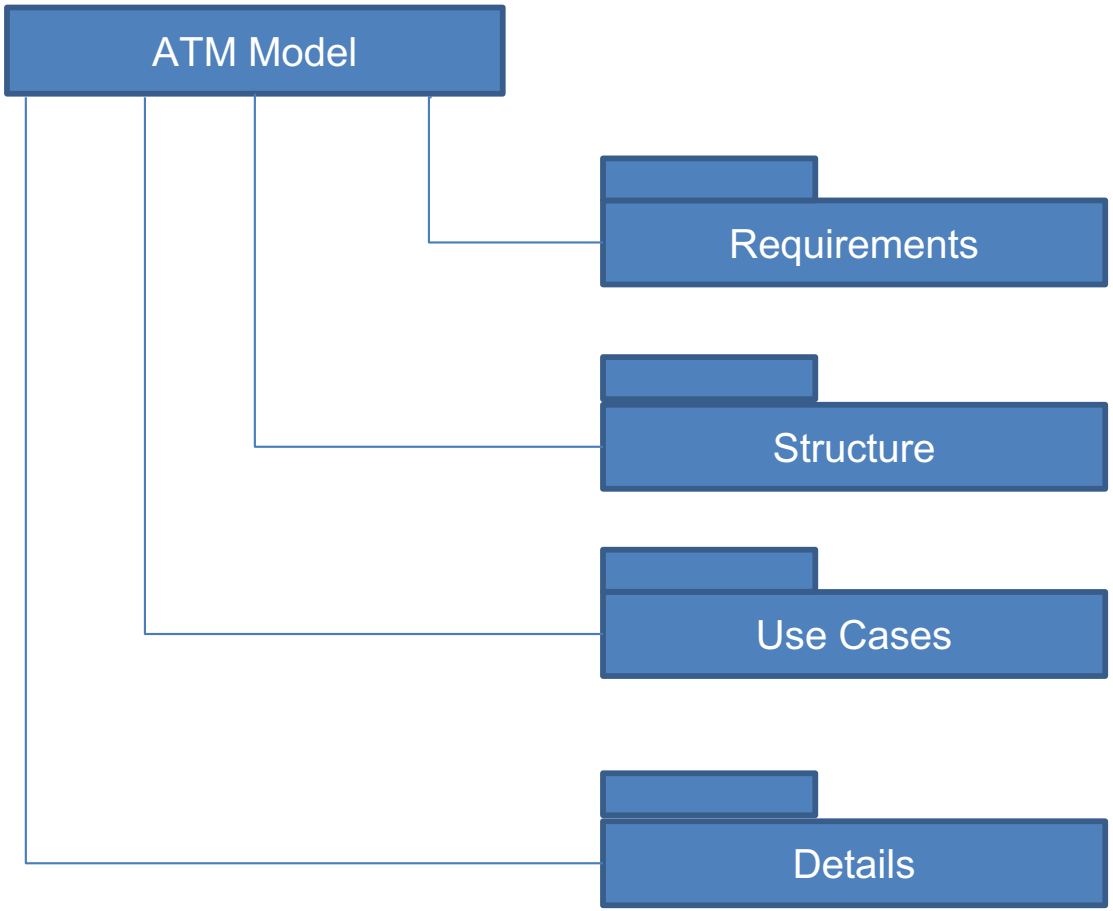


Note: This is just a sample diagram done without using a commercial SysML software. We just want you to get key ideas without getting too deep in syntax.

Massachusetts Institute of Technology

Step 4: Develop the ~~Behavior or~~ Structure Diagram

Here, we provide a sample package diagram for an ATM system. This is a basic high level package structure. You can elaborate it greatly to match your model/needs.

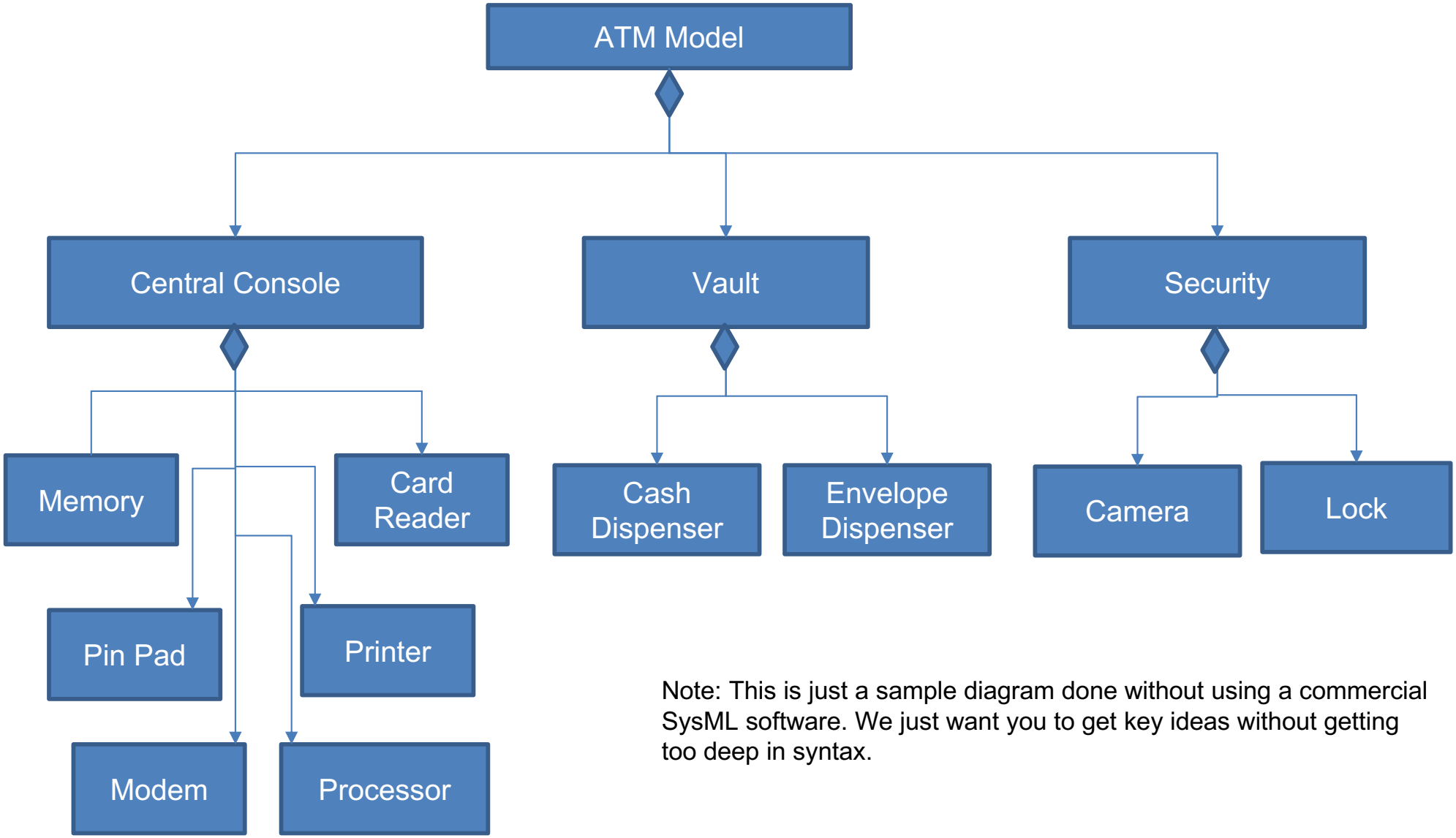


Note: This is just a sample diagram done without using a commercial SysML software. We just want you to get key ideas without getting too deep in syntax.

Massachusetts Institute of Technology

Step 4: Develop the ~~Behavior or~~ Structure Diagram

Here, we provide a another sample of structure diagram (block definition diagram) for an ATM system. Remember, you need to do only one structure diagram.



Note: This is just a sample diagram done without using a commercial SysML software. We just want you to get key ideas without getting too deep in syntax.

Massachusetts Institute of Technology

Critiquing an MBSE Approach

Overview

Critique this project MBSE approach covering:

1) Scope and Purpose (limit 300 words)
2) Strengths and Weakness (limit 300 words)
3) Qualities of Great Models (limit 300 words)
4) Conclusions and Recommendations (limit 300 words)

You should presume that the rationale of the critique is **"Evaluate whether a project should adopt the MBSE approach or stay with the status-quo.**"

Please note that this project combines weeks 1 and 2 project submissions – good practices– from a previous course so you will not be conducting a critique of a full model.

REQUIRED STEPS

Step 1: Critique this project following the instructions.

Step 2. Save and upload the project in the platform.

Step 3. Peer review the two critiques from your peers that the platform will automatically assign you.

Critiquing an MBSE Approach

1) Remark on whether the model is meeting the intended scope and purpose *[Limit 300 words]*

The MBSE approach applied to the ATM software project has a defined scope and purpose at a high level. From a critique perspective, the MBSE model provides a first time user with enough information to understand but leaves unanswered gaps. Without further information, it is not possible to fully assess the use of this MBSE model for the sister project.

The MBSE model is to be used for the software development of an ATM system. This involves the User Interface (UI) and the back end operations. The main purpose of this MBSE approach is to validate user requirements, maintain a single source of truth and facilitate information across all teams. Additionally, the requirement diagram is well structured and allows for any user to understand the stakeholder needs at a functional, security and performance level. The use case diagram clearly depicts the operator, customer and bank employee and their interactions with the system. Finally, the queries are clear and align with the purpose of the MBSE approach for this project.

While the previous information is clear, and it provides an abstract understanding of the model. I believe the MBSE model would benefit from the following.

- It is unclear under what conditions the model will be functional.
- The tolerance for errors is depicted in the purpose as “pretty low”. Given the importance of this requirement, I think this should be specified and not generalized. Moreover, this requirement is not clear in requirement diagram.
- It is unclear if this MBSE model will be applied to a specific banking system of a financial institution or if it’s attempting to encompass a general approach. I would expect the scope of this approach to be a single institution.

Massachusetts Institute of Technology

Critiquing an MBSE Approach

2) List the model's strengths and weaknesses. Your findings should relate back to specific instances of the model with screenshots or callbacks.

Strengths
(Limit 150 words)

- The MBSE approach was implemented in order to address several challenges listed in step 1 of week 1. If correctly developed, the model would be able to address them and greatly help the developing team.
- Applying MBSE to a highly regulated environment with strict requirements greatly increases its impact.
- Implementing web services, restful APIs and webhooks will allow the different teams to use design patterns and facilitate communication flow.
- The structure of the model diagram is well organized and can be easily understood by any user.
- Use cases include maintenance, operation and customer.
- Queries provided relate to stakeholder requirements and will allow the team to access valuable information.

Weakness
(Limit 150 words)

- The model library relies on translating MBSE information to the various software tools used by the different teams. Specifically, the translating requires a set of translation rules which would need to be updated manually. This seems to put the MBSE approach and the project at risk.
- Failure modes are not depicted. For instance, in case there is an issue with the MBSE model could we run a query? This relates back to under what conditions is the model functional.
- The validation example in step 2 of week 1 mentions, "check if all the components of the system are performing at least one task after the change is made". This could result in a livelock error where all components are in constant change in regards to one another while not making any progress in the task.

Critiquing an MBSE Approach

3) Evaluate the model against three qualities of great models, as well as supplementary behaviors or qualities you believe are relevant to MBSE. *[Limit 300 words]*

- **Availability of interface**. By implementing software industry standard communication protocols such as restful APIs, the team is ensuring loosely coupling, simplicity and reliability of the interfaces. If the model is implemented as depicted, the quality will be present in the MBSE approach.
- **Internally consistent**. While the structure diagram, purpose and scope provide an internally consistent design for all team members and stakeholders, the MBSE model seems to rely on a set of translation rules for the different software teams. Without further information on how these rules will be maintained and developed, it is unclear if this quality will remain true throughout the life of the model.
- **Verification and Validation with Models**. N/A--More information is required to reasonably judge this quality. I agree that this is an important quality that should be present in this MBSE approach, especially given the nature of the industry. However, not enough information is provided to assess the quality in this MBSE approach.

Critiquing an MBSE Approach

4) Offer your conclusions and recommendations, given the specific rationale for your critique. *[Limit 300 words]*

The MBSE approach for the ATM project has a large upfront cost given the approach. It requires investment to develop and configure communication across different software used by different teams. However, the benefits are clear and will greatly help the project achieve the stakeholder needs and user requirements.

For the sister project, this MBSE approach can be useful, especially if the upfront costs of rules and integration has been developed. I recommend that the MBSE team provide further information in the management plan given the lack of information. If the team is able to overcome the weaknesses in Step 2 above, I would be able to further asses the adoption of the current MBSE efforts.