Extended Definition Assignment

ENGL 361

Cody Smith

Rhetorical Situation:

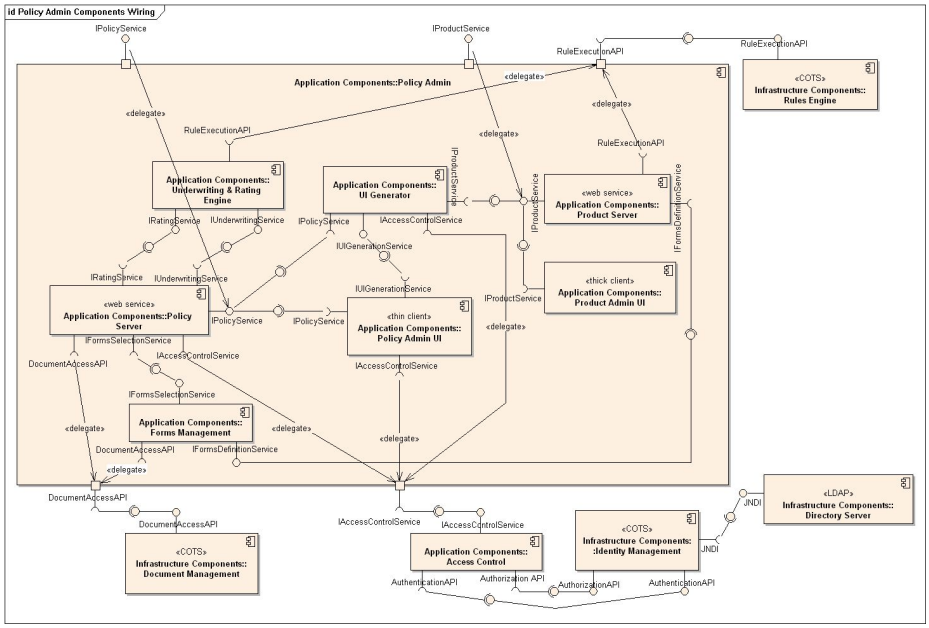
UML is a modeling language that is used in Software Engineering. UML 2.0 describes two types of diagrams, structural and behavioral which allows for visually displaying a design, and is especially useful in Object Oriented Programing. Given that during software development many different parties may have to understand or reference a design, having an organization use and understand UML can extremely useful tool. Given that many people involved in software development have different backgrounds it is important that if an organization uses UML to have the entire organization be aware of how to read the diagrams and why they should be making them when designing new software. In this fictional situation I am writing a letter to Management at Microsoft to introduce them to UML Diagrams. I aim to explain to them why they should be encouraging their employees use them when creating software and at a very high level how to read them for their own use.

Dear Management Team Members,

I'm Cody Smith, a software developer on our Microsoft Developer CORE team. I am tasked with teaching you about two types of related diagrams we are implementing for documentation, and will be utilizing in various level of our application design, called UML. Given how most of our applications are based in the Object Oriented Programming (OOP) paradigm these types of diagrams which you might not be familiar with will make designing and displaying our object hierarchies much easier, and I know we will find a lot of value with them.

UML stands for Unified Modeling Language, and is a more generally a tool for representing a design in software engineering. It's been around since 1997, and in 2005 was approved as an ISO standard. Specifically what we are interested in about UML is two types of diagrams that UML can describe called Structure Diagrams and Behavior Diagrams.

Structure Diagrams are a way to visualize the high level structures that go into a software design. They are typically written by our software architects (SAs) and will give you a high level view of what is involved in a software design. This can be further broken down into a component and class level. You should focus mostly on the component level, as the class level is really only useful for specific implementation details for the engineers. A component level diagram may look like so:

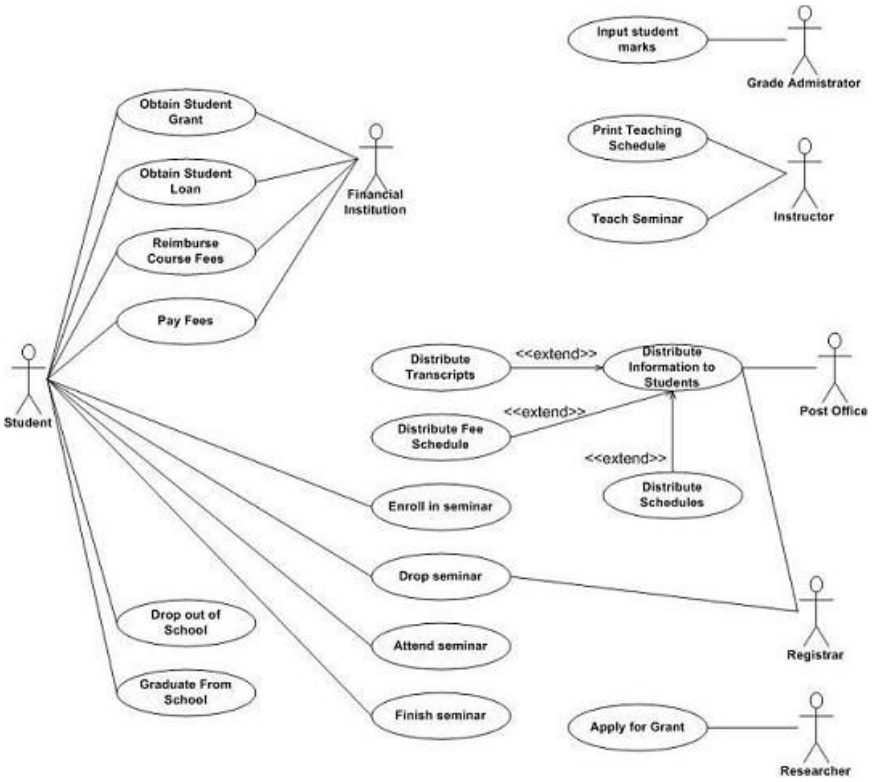


Structure Diagram: Component Level[1]

In this diagram, notice how you have blocks that are connected to each other with named connections. These blocks represent components in the software, and the connections how they are connected together. There is a lot more detail in these diagrams, but mostly you can

view this diagram as a roadmap of what components are in a piece of software and how they connect to each other.

Behavior Diagrams are a way to visualize what is happening in the software system. This can be further broken down to Activity Diagrams, Use Case Diagrams, Sequence Diagrams, and Communication Diagrams. Of most interest to you is the Use Case Diagram. The Use Case Diagram is used to show how a user interacts with a system. A use case diagram might look like so:



Behavior Diagram: Use Case[2]

The stick figures represent users, and the bubbles actions. This type of diagram can be very valuable to quickly follow the path of a user using an application and what they are expected to be able to do. You can use this type of diagram to see what and how your users are expected to use what your team is developing.

There is much more to UML Diagrams, but hopefully now you have a basic understanding of two types that you can benefit from. These diagrams are useful at all levels of our organization, from giving yourselves a high level understanding of the application design down to all the details of implementation for the engineers and quality assurance staff. Although they are very commonly used in academia, our organization has seen inconsistent use of these diagrams during the design and implementation process. No matter what product line you are involved with, you should build time into your schedules for your team to create these diagrams. You

should also encourage them to keep them up to date when updates are made, so that we can all benefit from the advantages of good documentation.

Thanks,
Cody Smith

Image Credits:

[1]https://upload.wikimedia.org/wikipedia/commons/b/b8/Policy_Admin_Component_Diagram.PNG

[2]http://agilemodeling.com/images/models/useCaseDiagram.jpg