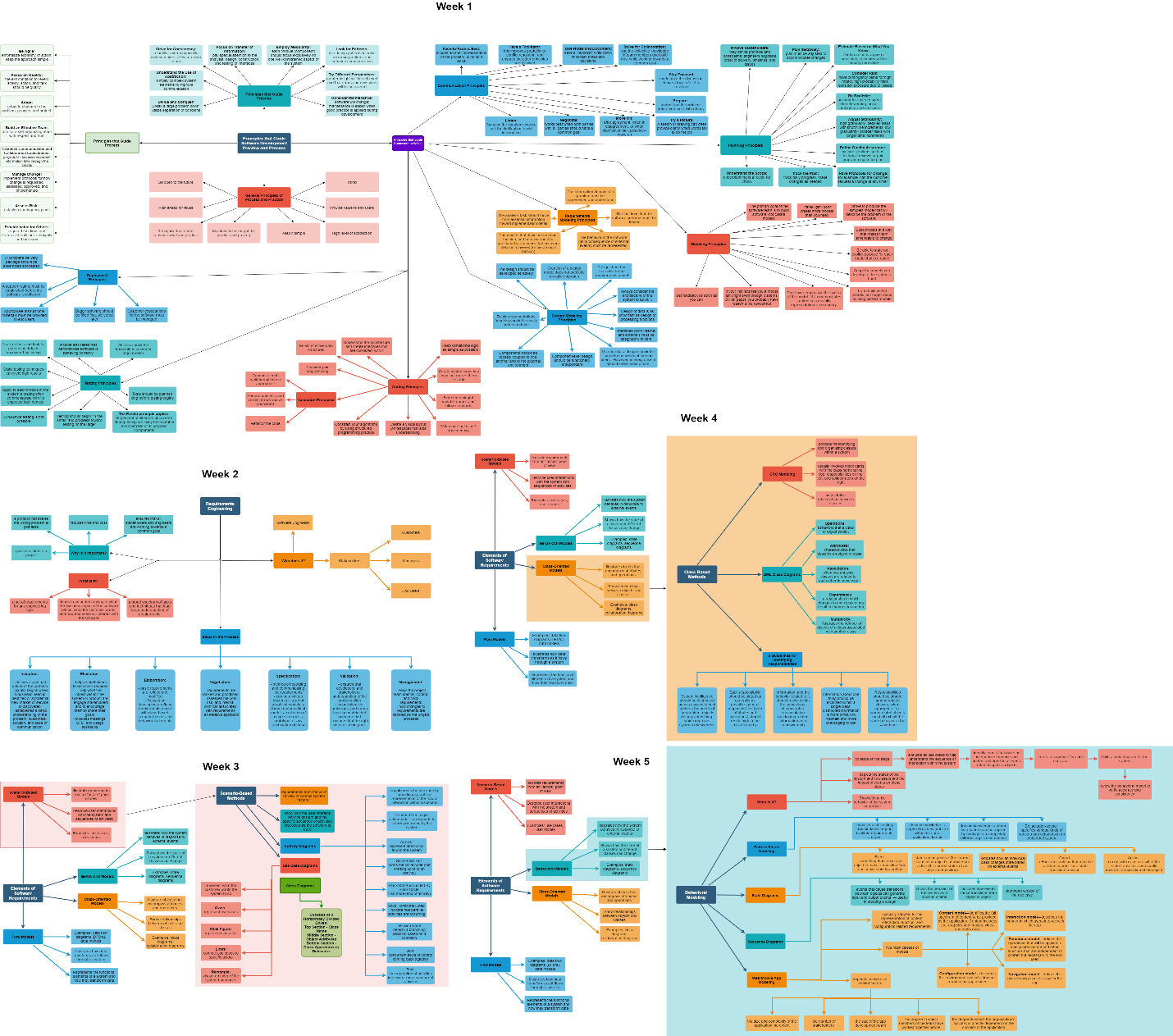
**Week 5 Assignment and Final Project**

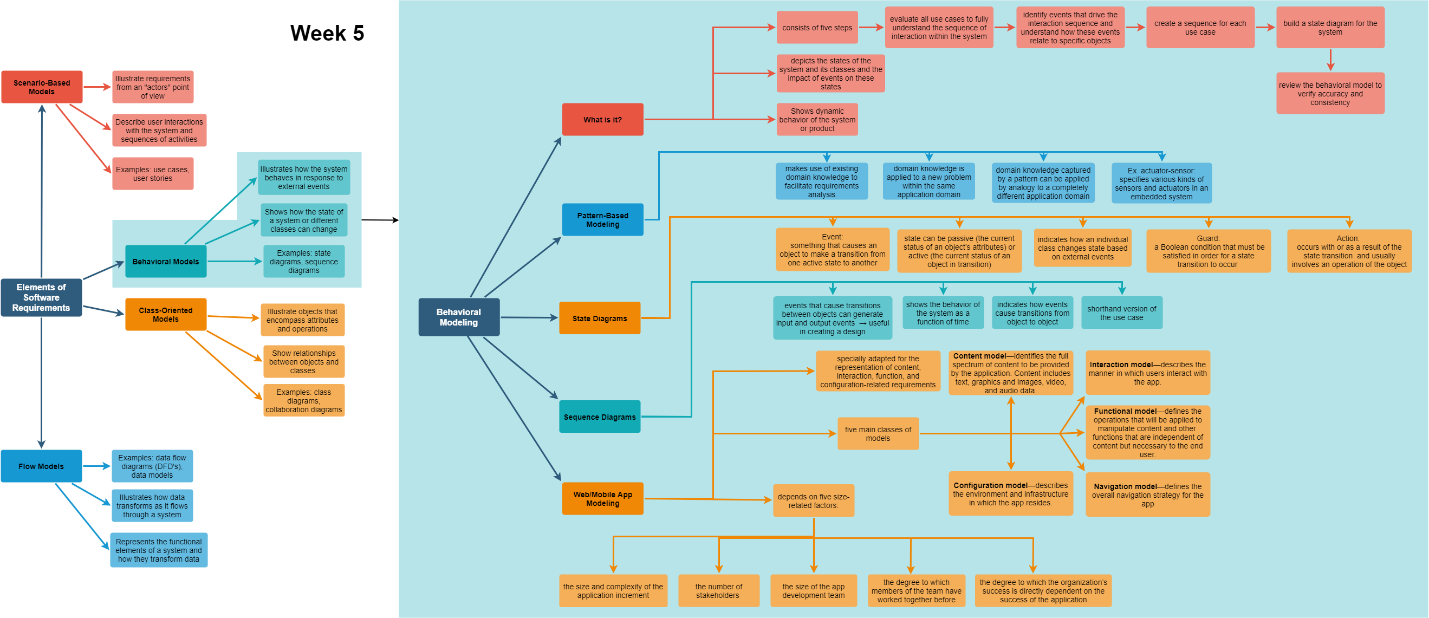
**Shaun Hoadley**

**CST 304: Software Requirements & Analysis**

**Professor Robert Key**

**April 12, 2021**

****

****

Requirements modeling of behavior gives stakeholders a way to visually identify how a system is going to react under specific circumstances. Behavior patterns in requirements modeling are kind of a shortcut for system design. Why reinvent the wheel if you have a library that has similar functions to what you are looking for. The purpose of modular programming or object-oriented programming is to create code that is reusable, and it is this reusability that allows us to begin to see the patterns. These patterns often cross over to multiple domains. In mobile/web apps, requirements modeling is specially adapted for the representation of content, interaction, function, and configuration-related requirements and relies on five size-related factors.

The various components of requirements modeling consist of scenario-based models, class-based models, behavioral models, and flow models. Each type of model shows a system from a different perspective allowing the development team and other stakeholders to see a more complete picture of how things should work together to address the problem the software is being designed to handle, generally even before any code is written.

Upon entering the software development life cycle, or SDLC, a core set of software development principles should be used to guide us. It is these principles that establishes a foundation to guide a software team while it is performing framework and umbrella activities, navigating the process flow, and producing a set of software engineering work products. Serving as shared rules and values for the stakeholders and the development team for a common goal, these principles allow the software engineers to “analyze a problem, design a solution, implement and test the solution, and ultimately deploy the software in the user community” (Pressman and Maxim, 2015).

According to Pressman and Maxim, there are a number of general principles used in software development. These include:

* Keep it simple
* Provide value to end users
* Employ a high level of abstraction
* Maintain the vision of the project and product
* Recognize that others consume what you produce
* Be open to the future
* Plan ahead for reuse
* Think!

(Pressman & Maxim, 2015). These principle are very useful, however, they are very high-level and can be broken down further into principles guiding process and principles guiding practice as shown in the full concept map above. The principles that guide the software development process are

* Be Agile
* Focus on Quality
* Adapt
* Build an Effective Team
* Establish Collaboration and Communication Expectations
* Manage Change
* Assess Risk
* Provide Value to Others

It is challenging, to say the least, to provide products valued by stakeholders if the requirements are full of omissions or are ambiguous (Pressman & Maxim,2015). In addition to the principles that guide process, are the principles that guide practice. According to Pressman and Maxim, these include the following:

* Divide and Conquer – break larger problems down and employ separation of concerns.
* Use Abstraction to Simplify Complex System Elements
* Be Consistent
* Focus on Information Exchange – pay special attention to interface design and implementation
* Focus on Modularity – every project component should be one well-defined part of a system
* Look for Patterns – pattern recognition reduces work for recurring problems and communication enhancement
* Approach the Problem from Different Perspectives
* Develop with Software Maintenance in Mind

(Pressman & Maxim, 2015).

I have attempted in my concept map to demonstrate how each component of requirements modeling works, both independently and with each other, to identify the problem to be solved, meaningfully deal with the said problem, and discover the potential pitfalls that may need to be addressed along the way.

**References**

Pressman, R. S., & Maxim, B. R. (2015). [*Software engineering: A practitioner's approach*](https://ashford.instructure.com/courses/81372/modules/items/4111209) (8th ed.). Retrieved from https://www.vitalsource.com