David Lambertson

**Alternative Software Life Cycle Models Summary**

**Important Points from Article**

* “Instead, we define a software life cycle model to be a reference model for a software development process…” [290]
* “While a life cycle model is insufficient to represent a definition of a software development process, or to describe the methodologies applied for software development, it does serve as a reference model for these processes and methodologies.” [290]
* “Thus, the requirements phase is often called user needs analysis, system analysis, or specification; the preliminary design phase is often called high-level design, top-level design, software architectural definition, or specification; the detailed design phase is often called program design, module design, lower-level design, algorithmic design, or just plain design, etc.” [290]
* “Much of the motivation behind waterfall life cycle models was to provide structure to avoid the problems of the "undisciplined hacker" [6].” [291]
* “The rapid, throwaway prototype, made popular by Gomaa and Scott [8] in 1981, focuses on ensuring that the software product being proposed really meets the users' needs.” [291]
* “Incremental development [9] is the process for constructing a partial, but deployment-ready, implementation build of a system and incrementally adding increased functionality or performance.” [292]
* “Evolutionary. prototyping extends the concept of incremental development to its ultimate conclusion, viewing the software life cycle as a set of numerous prototypes that are evolved through successive experimentation and refinement to meet the user's needs.” [292]
* “…reusable software is the discipline of attempting to reduce development costs by incorporating designs, programs, modules, and data into new software products [13.].” [293]
* “Automated software synthesis is a term used to describe the automated transformation of formal requirements specifications into operational code [15].” [293]
* “For every application beyond the trivial, user needs are constantly evolving.” [294]
* “When using incremental development, software is deliberately built to satisfy fewer requirements initially, but is constructed in such a way as to facilitate 'the incorporation of new requirements and thus achieve higher adaptability” [297]
* “The life cycle model evaluation paradigm (12) provides insight into how we might define, select or adapt a life cycle model to improve our process. 298 Currently, many project managers make this selection based on fuzzy perceptions and past experiences or blindly follow life cycle standards.” [298]

**Things I Didn't Agree With**

“The use of a rapid throwaway prototype early in the development life cycle increases the likelihood that customers and developers will have a better understanding of the real user needs that existed at time t0.” [296]

I disagree with this statement because I think it is better to have a basic system setup for users to test and give feedback on but then continue with that basic system instead of having a throwaway prototype. Creating something to just throw it away seems like a waste of work. Even though it may not be up to par or what is needed, you can take the first system that you create and just adapt to it based on the users needs. Through this process you are able to get an understanding of what users need and want, while not wasting precious time and resources working on something just to put it out with the waste.

**Things I Did Not Understand**

**Page 294 the automated transformation**

Something I had a hard time understanding was the automated software synthesis. I was not quite sure what was trying to be displayed and explained here. From what I think I understood, it's a process where we have something similar to an A.I. which creates all the program for us.