Daniel Carton

January 26, 2016

Dr. Gregory Gay – CSCE 747

The 10-Minute Test Plan

The main reason a company should inherit author James Whittaker's plan for testing is because it is quick, concise, and prevents wasted effort. The goal is to divide a plan using an application's attributes, components, and capabilities (ACC). By deciding the key adjectives, nouns, and verbs, a team can quickly decide which ideas overlap and should be prioritized during planning.

The suggested route for coming up with attributes has four tips that should be applied to the overall process of ACC. Simple, accurate, short, and to not get stuck on one thing. The first point is most important and encompasses the third and fourth tip. For example, if an application's attribute is to be *secure*, than the team in charge of that should know the constraints and potential problems to anticipate. They are crafty enough to figure out what needs to be done based off that one simple word. If you struggle to write down an attribute, component, or capability in a few words then your focus is being spent ill manneredly. Therefore you should strive to keep the requirements in mind and a proper description of your application. Overall these four things will achieve a company's goal of software testing in a timely manner resulting more time to meet precious deadlines.

When the author tested his group of Google engineer's, it seemed that they missed a key component of what he was expecting. The third guideline he promotes is to *guide a tester's thinking,* and it is one of the easiest recommendations of ACC to relate to. The summary of this advice is that you want a quick reminder that will lead to a functional thought or test. We all do this in classrooms and seminars when we jot down notes. The engineers seemed to make things too difficult and could not meet the 10 minute time constraint. Similarly to how the Google engineer's ended up using bullets, it's also how students take notes. I currently work in a lab, SysEDA, where we constantly make bullets on white boards as to what needs to be done, what problems are coming up, and what each software engineer is working on. Normally these notes are one word and self-explanatory. The idea is that it takes too much time to write every painstaking word that can be assumed by writing something short like **SEARCH** in big bold letters. An engineer could guess that he has to link a textbox to a database, list, or other. By using bullets and focusing on things that warrant attention the point of "no fluff" comes easily. In conclusion these helpful __ should help the tester think fluently and logically on the concepts at hand.

The only part of the paper that is distracting to me is distinguishing components before establishing capabilities. When personally trying to relate this reading to my experiences at work, capabilities came naturally after attributes. I thought "you can import 3d ships" or "you can map out a mission across the seas." Instead of the nouns that components desires, it was

easier to think of what the application actually does. Obviously Whittaker isn't implying there is a set order of operations. Introducing the inquiry of what a user can do is easier to answer than what a user has at hand – which is essentially what capabilities and components are, respectively.