Ushahidi Pre-Read
Case 3 SI 310 Fall 2019
By John Leslie King

Ushahidi. the third case for 310, is software to support crowd-sourced mapping and analysis using information from social media and other sources. It was developed over ten years ago (read the stuff on the link above) for political application and extended to address humanitarian emergencies and other challenges. It is and depends on other infrastructure (*e.g.,* telecommunications, reliable electricity). It can heighten or hamper information inaccuracy. The case is "unofficial," written by three former UMSI students to show how well-intentioned aid workers using Ushahidi for humanitarian purposes encountered infrastructure problems before Ushahidi was appropriated by PART-3 terrorists.

The case extends the stakeholder and infrastructure discussions of 9/11, adding requirements. The good analyst knows things can go wrong. Perrow says that normal accidents cannot be prevented, but good analysts anticipate them better than others by looking where others don't look. Most stakeholders do not appropriate infrastructure and turn it toward bad ends, but some do with huge costs like 9/11. United 93 shows how the good guys can turn the table on the bad guys. Stakeholders and interests change over time. Determine what infrastructure is involved (use infrastructural inversion if necessary).Connect infrastructure to stakeholders and interestse. Agents determine whether stakeholder interests are to be served by the infrastructure, are to be ignored by the infrastructure, or are to be thwarted by the infrastructure.

Ushahidi introduces requirements. The concept of Ushahidi can be useful. Many humanitarian aid organizations now use some form of map mashups in their aid efforts. The trick is to get it right soon (fast). Figure 1 from the Syllabus includes requirements. at the intersection of stakeholders, interests, context, solutions. The agent must determine the right requirements. This is hard to do but important. Search on <requirements>. Now search on <"clear unambiguous requirements"> (include the quotation marks). The latter yields many hits, but in 40 years I have yet to see "clear unambiguous requirements" for anything more complicated than a thermostat. People often make unworkable demands as requirements (*e.g.,* "I push this button and all my work gets done…" or "it just has to be awesome…"). Requirements analysis is hard because there is always ambiguity. That's no excuse. Remember the third umpire: "They ain't nothing 'till I calls 'em."

Assignment: Write no more than 350 words. Explain the stakeholders in Ushahidi and their interests. Describe the information infrastructure of Ushahidi. Where does information important to Ushahidi come from? What problems did the aid workers have with Ushahidi? If you had the job of developing requirements for something like Ushahidi, what would you include given the 310 case?

Rubric: I will discuss this in lecture. Apply what you learned in 9/11. Develop the skills yourself. You know the drill. Read the background readings on requirements, the "official" Ushahidi case from the Ushahidi web site (opening link),

the Ushahidi case, and this pre-read. 9/11 heightened your senses about stakeholders and infrastructure. Ushahidi expands them to include requirements. Consider the fact that the Ushahidi software was developed for one purpose in one place, but applied to other purposes in other places. The *context* changed. This matters to stakeholders, interests, and infrastructure. Decisions have consequences. What decisions have been made with what consequences? Infrastructure works "as well as it can," but this is sometimes "not good enough." The infrastructure of Ushahidi was appropriated by PART-3 stakeholders. This was *not intended* by the Ushahidi designers, in the same way that those who designed air travel infrastructure did not anticipate 9/11.

The country of Torabia is fictional. Most of the Ushahidi case is fictional but it *could* happen. The agent must notice what is *possible* and ask whether it is *likely*. Unlikely things can happen with major consequences. For example, the Tylenol Murders of 1982 changed forever the way over-the-counter drugs are packaged in the U.S Those changes made the repeat of those murders unlikely. The agent must understand what others do not, even if the others outrank or get paid more than the agent. The Ushahidi case is not long. Read it more than once! Don't overcomplicate it. The lessons are clear once you spot them.

Cheap/Fast/Good is simple in principle. Once you become adept, you will be amazed at how routine it is. You'll wonder why others don't do it. It takes work and practice initially, but it gets easier as you get better.

Kinda Reading About Requirements

This section introduces *kinda reading. Kinda read* the two readings for this case. They give you some sense of requirements. The Bergman, King, Lyytinen paper discusses the political ecology of requirements engineering. The King and Simon paper discusses complexity in requirements engineering. We'll talk more about *kinda reading* in lecture. Too many papers have been written on requirements engineering to expect any BSI student to read them all. These provide an introduction. People often do not know what they want until they see what they can get, which makes getting requirements hard. Yet, as Ushahidi shows, getting requirements right is important. Parse the job to figure out whether there are requirements, and if so, what they are.

TL;DR stands for either Too Long' Don't Read or Too Long; Didn't Read. Either way, it is *not* an excuse for not readingl. Read everything in 310. We will tell you what to really read and kinda read. Kinda read the requirements reading (the Bergman/King/Lyytinen papers and the King/Simon paper). The material that follows walks you through kinda reading. Kinda reading saves time but conveys the gist. It gives you enough to know what's there, in case you need to come back and really read. The objective is to get enough understanding to be of use.

Here's how to do kinda reading. First do a quick scan for title, author, publication venue, and date. The paper titled, "Large-Scale Requirements Analysis Revisited: The need for Understanding the Political Ecology of Requirements Engineering suggests things when you parse the title and authors. Among the authors' names are Mark Bergman, John King, and Kalle Lyytinen. One name should

ring a bell. The paper appeared in a journal called *Requirements Engineering* in 2002. The second title is "Complications with Complexity in Requirements," by John King and Carl Simon (again, the bell). It appeared in a journal called *ACM Transactions on Management Information Systems* in 2015.

Reading nothing more we know that some people study requirements. They publish on the subject, and publications on the topic date back to *at least* 2002, was going earlier, and continues. You can get all this without having to read *anything*. Think about context. The syllabus says 310 deals with requirements. The instructor for the class was a co-author on two papers in the requirements area. Look at the abstracts. The first says requirements are political decision processes, mappings justified by concerns of principals (stakeholders), that solutions are socio-technical ensembles of non-linear behavior affected by domain complexity and political ambiguity. Solutions stabilize into specifications affected by organizational power. Requirements engineering brings together technical, social, economic and institutional factors. The second paper, written later, notes the difference between complicated and complex. Complicated can achieve successful solutions; complex often leads to failure.

Scan the text. One article says there are proponent and opponent coalitions, but naïve requirements engineering literature covers *only* proponents. The confusion over complex and complicated is evident in the first paper. It is confusing but it can be simplified: complicated problems might be handled, but complex problems seldom are. Look up more on requirements so you can tell an internship supervisor, employer, client, etc, that you know a lot about requirements. The person you are talking will perk up because they know that requirements analysis is at once *important* and *difficult* part of the system development life cycle. People blow it off, because it is hard. Getting good with requirements is a good way to work your way up.

Genuinely complex problems probably cannot be solved. No amount of requirements engineering will help. Simplify or give up *now.* Success has many parents; failure is an orphan. Consider all the stakeholders. Look for the opponent coalition (trust me, it's there for all real decisions). Think about what *will and will not happen,* irrespective of requirements. Recognize bad possibilities early and defended against them. Consider 9/11. Thinking only of proponents misses the terrorists! The Ushahidi case showed problems with infrastructure even when the system worked. PART-3 surprised everyone. A good agent anticipates such things.

People often talk about requirements as an unpleasant but necessary chore, and seek to get through requirements as quickly as possible. The good agent takes the time to get requirements right. When first encountering a job, ask "where are the requirements?" If there are none, the job is *already* in trouble. If they are problematic (there are lots of ways this can be the case), the job is *already* in trouble. Fix the requirements and get the stakeholders (at least those in the proponent coalition) to agree with the requirements. There might be trouble, but at least it is not inevitable, like when a project has nonexistent or bad requirements.