Blume
Software Engineering LLC

You have a big project underway. Your developers are all set with their tools, they know the implementation language(s), the server platforms, cloud or wherever the project will eventually be deployed when finally in operation. But, as seems to be the standard in the industry, already the project is lagging behind technical goals and schedules, there is a lot of re-work, lots of meetings and emails being sent in salvos between different individuals, groups and territories. This is the typical software development operation and complexity is feeding the chaos.

You start asking questions; "Why all this email? It's good that people are communicating, but what is being settled before it's built and why are we re-coding things again and again?"

I would be asking the following questions. How is the project lead communicating with the different development teams? Are people working AS a team or are different "silos" occasionally "launching missiles" at other "silos"? If they're not working as teams, what efforts to bind them into teams are the leads using? Is everyone working to the same set of requirements? What are those requirements and where are they being stored so everyone can refer to them? Is there a requirements model at all? Who is unable to utilize any existing requirements model due to lack of training/experience? Are development teams working to the same interfaces so that their finished "parts" all "fit"? Are requirements changes being immediately distributed so that everyone is on the same page at the same time?

All these questions have one thing in common. They are all, every last one of them about COMMUNICATION. They are about communicating requirements from stakeholders to the developers and between developers who should be working to the same set of requirements and to the same architectural design. Team leads are facilitators – who should be especially adept at the tools that help them communicate to stakeholders, from stakeholders and then to their respective teams – and between themselves.

More questions: What tools are designed to facilitate communication between stakeholders, project managers, team leads and developers? Which are best for YOUR piece of the project? Where are these communications stored and what form should they be in; text or graphics? How about both? Do you want your network design to be described in text – or would a network diagram be more immediately useful? What's the best way to capture, store and then communicate requirements? Are these specific technologies or practices – or something else?

Without effective communication, you are doomed to repeat the Information Technology version of Abbot and Costello's "Who's on first?" Obviously, you won't think it's funny if your approach to development is over budget and behind schedule, even if that appears to be an industry "standard".

https://youtu.be/kTcRRaXV-fg

What do you do? Well, take a look at your communications, first. That's the backbone of your entire effort. Look at your "process". Is Agile working for you? Are user stories capturing all that you need to communicate to your leads and your developers? The bottom line is the answer to this question; "Are user stories doing all that we need to do to capture requirements?" If not, then you certainly cannot communicate everything you need in order to fully-inform your "chain of communications". If user stories do all that you need, then use them. If not, ask yourself, “What's missing?” Then, if you find that Use Cases are more complete and help more thoroughly communicate stakeholder requirements, why are so many people saying that they're more difficult to write? I think you'll find the answer to that question to be one of “Do your 'requirements people' actually KNOW HOW and WHY to write Use Cases?” I find that the answer to this question is, “No, most people who think they are writing Use Cases to not understand what Use Cases are.” So, they cannot possibly write them on that basis and it is difficult for them because they don't understand them. So, User Stories are a stopgap, a “cheap” substitute for Use Cases – which are more difficult to write because they are so simple that most people cannot grasp the concept. They are over-complicating what is quite simple to write, but which has MUCH MORE detail than a User Story.” Do you have a glossary? Do you have little pictures of actors named “Who” and a picture of first base that “Who” is on? Such a simple diagram would reveal that the guy on first base is named, “Who”, ending the ambiguity inherent in the statement “Who's on first”.

http://www.seguetech.com/blog/2015/08/10/User-Stories-versus-Use-Cases-Pros-Cons

According to the article at Sequetech, “The cons: Use Cases are meant to provide such a formalized blueprint of the project, Future of CIO explains, that they often leave little room for negotiation or project additions. Additionally, states All About Agile, the Use Case can get a bit complicated and isn't a format that is generally palatable for end users or business people.” I have to disagree with that statement that it leaves “little room for negotiation or project additions”. The Use Case part of the requirements model should remain flexible up to the point where anything derived from that model interferes with the users' job, how they perform their duties. If there is a need for Business Process Reengineering, changing the way information workers do their jobs, that will necessitate re-work of any pre-existing Use Case Model. New procedures and workflows must be incorporated as needed. Remember that these processes are intended to facilitate your management of the project, not interfere with it, so you should do what works for you and your team(s). Tools should support communications, but the communications are the primary concern. So, communication, not tools or process – should drive your project. Where User Stories are sufficient or enhance the collection of detailed Use Cases, use them. When you need detailed requirements for your developers, you might need to do full-fledged Use Cases, and you will need to have someone who really does understand them and can write them in a timely manner. If the requisite communication ensues and requirements are completely fulfilled, your project is on track. Reworking code and software architectural design due to lack of communication – will set you back.

Adhering to one of the “process ideologies” for the sake of “purity”, should never be a goal, especially since each one of them are flawed in some respect. While RUP is a TOTAL and complete process, you have to consider if it is necessary for your project. If you are not building a software component that is mission-critical to say, sending astronauts to Mars, where lives are in the balance if your software fails, then you probably don't need something as thorough or as time-consuming and expensive. As Martin L Shoemaker, author of “UML Applied” (http://www.amazon.com/UML-Applied-Perspective-Experts-Voice-ebook/dp/B001KW00X4) says, “But my main message is that UML is about – say it with me, class – *communication*. If you can't understand the problems, then I've failed to communicate.” Only do enough to communicate completely – without repetition.

Choose the requirements-gathering approach you want – as long as it communicates what you need. Don't discount Use Cases because a few purists are saying that User Stories are better. They're better in some situations, but not in all. Sometimes, I even break out the CRC (Class, Responsibility, Collaboration) Cards. Or, I might use Martin L. Shoemaker's lightweight “Five-Step UML” approach, even though I'm familiar with RUP (Rational Unified Process). It all depends on the size and criticality of the project – or any particular PIECE of the project.

https://blogs.versionone.com/agile_management/2013/07/16/why-i-dont-like-user-story-templates/

Let's then talk about tools that facilitate communications. You can use post-it notes and pencils. Of course in a larger project, that might become unmanageable. Enter "Visio". You CAN do your user stories and/or Use Cases in Visio. I imagine you can do all kinds of diagrams with it. A tech at MS just reported to me from the MS Store that Visio now does class, sequence, database, Use Case, Activity and State diagrams. The details on code generation (forward engineering) and reverse engineering class diagrams from the code – remains unclear. Nor is there any detail on whether you can link specific use cases to activity diagrams and from those to class diagrams. I also read reviews on the latest version of Visio and would have to decline a recommendation on the product just on those grounds. But, if all you want to do is make a few simple, lightweight diagrams, you could probably get away with some more stable, prior version of Visio.

However, if you need a more robust tool that has all you will ever need, there are many options. IBM Rational Rhapsody Architect for Software is the latest from IBM/Rational, Rational being the outfit started by the "Three Amigos", Jacobson, Booch and Rumbaugh and seems reasonably priced. I haven't tried it, but might download the free trial at some point. I note that it has it's own coding environment and supports C++, C#, C and Java. From what I remember of Rational's old XDE and Rose, these products will do more than facilitate Use Cases and all standard UML. This product also does round-trip engineering.

I also used to use "Visual UML", which enabled you to do full round trip engineering, as well as Use Cases which could be linked to activity, then class diagrams for full traceability from end to end (from requirement to implementation through design/architecture). Unfortunately, Visual UML is no longer available, but I include it here because the owner/developer of Visual UML sent out a letter to all his customers informing them that he was going out of business and that his diagrams would be supported by Visual Paradigm. So, it is by way of Visual UML that I was introduced to my current and favorite of all software engineering tools, Visual Paradigm which even at the developer level has scads of diagrams and does complete round-trip engineering. You can generate code from the models (class diagram) and then work on the code and reverse engineer it back into the model and all these diagrams which capture requirements, design, architecture, implementation and deployment – will link at a fine level to provide traceability from requirements all the way to the specific server, server farm or cloud you've deployed the final product on.

In addition to User Stories and Use Case diagrams and detailed flows, Visual Paradigm has class, sequence, communication, state machine, activity, component, deployment, package, object composite structure, timing interaction overview UML diagrams.

Visual Paradigm also has Requirements Capturing diagrams, such as Textual Analysis, Requirement and CRC Card. Interface wireframes are also included: Android Tablet, Android Phone, Desktop, iPad, iPhone and Web wireframes are included.

In the Database Modeling section we have Entity Relationship and ORM diagrams. Business Modeling diagrams are also some of the many diagrams available, including SysML, SoaML, Impact Analysis and "Others" (Overview, Mind Mapping, Grid, Document and Brainstorm).

So, you get a COMPLETE set of UML diagrams plus many others and for less than you'd pay for Rational Rhapsody. Additionally, the Visual UML IDE integrates into the Visual Studio IDE.

More on this, later, as I continue to build out the http://www.blumesoftwareengineering.com web application. Stay tuned.