Isaac Delgado
USC ID: 4189182267

1. (24 points) For each ICSM principle, identify at least 1 potential challenge and supporting rationale for the following types of software development projects

Stakeholder Value-Based Guidance

Small scale, non-life-critical projects (less then 10 people, shorter than 1-year duration):

In small scale projects there is a lot potential change that can occur since through out the life cycle of the software given the small amount of people and short period of time. Thus one clear potential challenge is the need to accommodate new success-critical stakeholders that may come along the way. Additionally, it may be difficult to identify "true" success-critical stakeholders as the many of the stakeholder roles can be combined/reduced to an individual role given the size of the project. Thus having one too many success-critical stakeholder may be prominent in small scale projects.

Medium scale, moderate business-critical projects (less than 30 people, shorter than 3-year duration) (add here):

Medium scale projects will incorporate a little bit of both small and large scale challenges. Given the size of the project there is a high risk for change to occur and thus an addition of new success-critical stakeholders could join along the life-cycle of the software. However, the potential additional stakeholder will lead to an increase in difficulty of win-win negotiations. Thus it will be critical to continuously review with stakeholders to adhere to the potential changes and win-win equilibrium.

Large scale, safety/mission-critical projects (more than 100 people, longer than 3-year duration):

The clear issue with stakeholder value-based guidance at larger scale projects is negotiating and maintaining a win-win scenario between all stake holders. At this level of project there will most likely be a lot of stakeholders and the more the stakeholders the more likely it will be difficult to give everyone what they want. As the project proceeds through the life-cycle it will continuously be difficult to make sure all stakeholders are involved throughout every step and keeping everyones win-win scenario balanced.

Incremental Commitment and Accountability

Small scale, non-life-critical projects (less then 10 people, shorter than 1-year duration):

Given a small scale project I see a potential challenge of avoiding large commitments and/or completely disregarding the incremental process. Since this size of a project is small with a very short duration it may be tempting to set a total commitment with the project instead of carefully committing to small increments. Additionally, taking too big of increments versus total commitment can be just as fatal to the software project leading to a lot of rework and catching up towards the end of the software duration.

Medium scale, moderate business-critical projects (less than 30 people, shorter than 3-year duration):

A potential issue I see with medium scale projects with regards to incremental commitment and accountability is to account for changes in the scope of the project. Medium scale projects can shrink or expand in size with regards to stockholders, requirements, end-goals, etc. Thus making sure the project adapts to any potential changes by adjust the next incremental commitments will be vital to the project. If for instance the size of the project increases drastically, we must review every aspect to account for unforeseen risks and incorporate this new reviewed report into the next increment.

Large scale, safety/mission-critical projects (more than 100 people, longer than 3-year duration):

At large scale projects a potential challenge is to have an equal amount of incremental accountability between everyone and to keep everyone happy throughout the life-cycle. As the range of success-critical stakeholders are wide it is important to be keep the commitments small and steady. These small commitments may make it difficult to keep everyone happy as impatient stakeholder will likely want larger commitments. Thus considering the level of accountability at each commitment is crucial at this level of project.

Concurrent Multidiscipline Engineering

Small scale, non-life-critical projects (less then 10 people, shorter than 1-year duration):

The potential challenge with small scale projects regarding concurrent multidiscipline engineering is similar to that of incremental commitment and accountability. Given the small size of the project it will be temping to not only make large commitments (if not total-total-commitment) but to also take a sequential approach to developing the project. With such a short duration an agile methodology is easy to gravitate towards and thus ignoring concurrent development may happen naturally.

Medium scale, moderate business-critical projects (less than 30 people, shorter than 3-year duration):

A potential challenge I see with medium scale projects may be the ability to concretely define all developmental activities that will need to be worked on in a concurrent manner. As this scale of project has a potential of growing or shrinking it may be an issue too make sure all developmental activities are defined so the project has an efficient concurrent work flow towards the win-win outcomes defined. In contrary with larger scales you can slightly benefit from having too many developmental activities cine there is a longer duration and more chance of risk, where as in medium scale the schedule is tighter and budgets are most likely smaller.

Large scale, safety/mission-critical projects (more than 100 people, longer than 3-year duration):

For large scale projects a potential challenge regarding concurrent multidiscipline engineering is to continuously develop in a concurrent manner. This principle illustrates the “hump chart” discussed in book, and according to the hump chart we must always be spending some level of activity on majority of the developmental activities (excluding some like Monitoring and Control during the beginning exploration phase). With so many developmental activities to account for such as “Envisioning opportunities”, “System Scoping”, “Requirements”, “Architecting and Designing solutions”, etc. it can be difficult to make sure all categories are kept up with through a large scale project. For instance “Requirements” will change over time and with a larger project it will inevitably change and will require a lot of attention to make sure all areas of the project are accounted for.

Evidence and Risk-based Decisions

Small scale, non-life-critical projects (less then 10 people, shorter than 1-year duration):

With small scale projects the evidence and risk-based decision may be heavily overlooked and event-based or scheduled-based strategies will be tempting to adopt given the short duration of the project. Since the range of components will be narrow for a small project it is easy to gravitate to the agile method which will entail a sequential development style. This will naturally lead to event and schedule based decisions in order to get the project done by the pressures of the short deadline. Thus risk-based or feasibility analysis could be easily forfeited for scheduled priorities in a small scale project, which will violate the fourth principle leading to a potential failure.

Medium scale, moderate business-critical projects (less than 30 people, shorter than 3-year duration):

With medium scale project there can be pressures from the size of the project and the schedule allotted that must be balanced. This of course can happen in small or large scale projects as well but in medium scale it may be more likely to happen since these projects are in between small and large scales implying it could shrink or expand in projects size. Given this pressure, a potential challenge I see regarding evidence and risk-based decisions is the temptation, at some point, to “build it quick tune later” approach. The scheduled pressure may cause a sense of urgency and deciding that a project developed now with many risks is better than a relatively risk free project developed later. This of course could lead to a bunch or re-work.

Large scale, safety/mission-critical projects (more than 100 people, longer than 3-year duration):

In large scale projects a potential challenge when making evidence and risk-based decisions can be developing solid feasibility and risk analysis reports to support decisions. With larger size projects there may be a lot of different components and stakeholders to account for. Thus a more complicated project/system will require a more in depth review to get a clear feasibility and risk based analysis. This principle will closely follow the third principle, concurrent multidiscipline engineering, as the evidence and risk-based decisions will accommodate and synchronize all concurrent activities. This further drives my point that the more complicated and larger the project the more detailed and robust the the risk analysis will be.

2. (10 points) The ICSM suggests 5 life-cycle phases, use EP-01 Top software failures in recent history to identify specific examples of feasibility evidence that you can use to reduce the cone of uncertainty in each phase. In addition, provide support rationale how the evidence would reduce the uncertainty accordingly.

Exploration Phase:

O2 suffered a big set back after a glitch prevented customers from using 3G and 4G services. The glitch was brought on from using Ericsson equipment. This is a perfect example of Feasibility evidence for the exploration phase as during this phase we are looking at all potential solutions which entails looking at different software NDIs, Greenfield/Brownfield development, requirements and goal running, etc. In this case the equipment from Ericsson causing the glitch will be taken into consideration during the exploration phase and maybe other competitors (like Qualcomm, Infosys, Avaya, etc.) can be evaluated to be used instead. This will surely reduce the cone of uncertainty even though at this phase uncertainty is at its highest.

Valuation Phase:

After Facebook acquired Instagram and WhatsApp there has been connected outages between all three platforms, seemingly if one falls they all fall together. The issue was claimed to be caused by “routine maintenance”. This could be used as feasibility evidence for the valuation phase. During the Valuation phase we must decide whether the chosen solution or chosen next-step from the exploration phase is suitable by evaluating the risk further. In this example Facebook could have evaluated the idea of acquiring Instagram and WhatsApp as well as evaluate their architectural connection to each other during this Valuation phase. This could have shed some light on the risk of one system failure causing the others to fail as in this example and reduced the cone of uncertainty at this phase.

Foundations Phase:

During the British Airways IT glitch in 2019 that disrupted the flow of flights, the online check-in system and a system for departing flights were effected causing the airport had to go back to manual check-in procedures. This could have been great evidence during the foundations phase of their software cycle as they could have tested for the case of online check-ins failure and developed accordingly. This risk analysis I think would have been high but addressable and they would have came to a smoother back process, thus reducing the cone of uncertainty.

Development Foundations Phase:

During the Wales NHS system failure the hospital was unable to execute day to day activities like access patient files, contact patients, or save patient notes. It was reported that the issue was due to a technical short coming and not a cyber attack. This is great feasibility evidence for the Development Foundations Phase. During this phase there must continuous testing on the system before the actual deployment and fully operations phase. If proper testing was done then this issue may have come up earlier and addressed during the risk analysis before making its way to operational status. This issue, if caught during this phase, would have made great feasibility evidence and reduced the cone of uncertainty at this phase.

Operations Development Foundations Phase:

The Nest Smart Thermostat had a major glitch causing the thermostats to lose temperature control. It was stated that this glitch was caused by a software update along with some air filters and incompatible boilers associated with specific users' homes. This is good feasibility evidence for the Operations Development Foundations Phase as during this phase the product/software is operational and now must be further evaluated and maintained for more enhancement and tuning. In this case Nest Smart Thermostat was fully operational and they were updating the system but ended up breaking it in the process. However this update process led to the next software update which subsequently fixed the issue. Because of the continuous maintenance they found some feasibility evidence and handled it accordingly, thus reducing the cone of uncertainty. This iterative discovery of risks will inevitably happen to major systems and the reason further maintenance and analysis is crucial during this Operations phase.

References

Jee, C. and Macaulay, T. "Top Software Failures in Recent History." Computerworld. August 2019. https://www.computerworld.com/article/3412197/top-software-failures-in-recenthistory.html

Boehm, B., Lane, J.A., Koolmanojwong, S., and Turner, R. *The Incremental Commitment Spiral Model*. Addison-Wesley, 2014