Theory 1:

Scale Factors:

PREC –

Precedentedness: A metric of how unique the project is in the history of the organization that is creating it. Put another way: Have we ever done anything like this before?

Story: Management wants to develop a product for the web. This is the first web based project they have ever developed. They have also never built software that was used by more than 3 people, but this project will be used by several million. This is a situation with very low PREC.

FLEX-

Development Flexibility: A metric for how rigid requirements are for a specific project, and if there is a premium on finishing early.

Story: Management wind a government contract for the company, but the project must meet a very strict set of requirements that include a very specific interface. Additionally for every day before the deadline the team delivers early, the team will get $10,000. This is a situation with very low FLEX.

RESL-

Architechture/Risk Resolution: This metric consists of factors involving the organization's capability for planning for and dealing with risk. It also deals with the how much time is set aside for architecture and how skilled the architects are that are involved.

Story: Management decides that risk management and architecture is too expensive , so they forbid the team from doing either at all. They have thrown their entire budget and time into programming the project as fast as possible. The unfortunate thing is that there is one huge risk factor for the project that could destroy it's commercial potential. This is very low RESL.

TEAM-

Team Cohesion: This metric deals with how well the various groups involved in a particular project work together, it also captures the amount of experience they have working together. This includes everyone involved with the project: clients, programmers, end users, etc.

Story: Management wins a contract working for an evil dictator to prepare his website for the upcoming election that he will definitely win because he's a dictator. Everyone in the organization hates the dictator, and he is terrible to work with since everything has to be his way. Teambuilding exercises failed, because he put everyone on the other kickball team in prison. This is very low TEAM.

PMAT-

Process Maturity: This metric uses a fairly complicated measurement technique involving the Sotware Engineering Institute's Capability Maturity Model, or CMM, and it's Key Process Areas. All of the key process areas involve a determination of how often an organization follows a set of best practices.

Story: A new startup opened in Raleigh. All management is done by the CEO who is one of the four employees of the company. There's no real management of any kind going on at all and their process consists of programming constantly to try to finish each project in the pipeline with barely enough testing to get it out the door. This would be an organization with very low PMAT.

Downsies:

RELY:

Required Software Reliablity: This is a metric for what actually relies on the software's success or failure. If a piece of software is relied on for whether people live or die, like in avionics RELY is very high.

Story: Management wins s a contract to work on the software for the life support system of the new space station that china wants to build. Since if the life support system fails people die, this is a situation with very high RELY.

DATA:

Database Size: This metric is arrived at through a calculation of database bytes divided by lines of code. The idea is that if there is a large database there will be a large amount of effort required to generate test data for the system.

Story: Management wins a contract with SETI to work try to find patterns in their data. Per http://setquest.org/join-the-quest/data-api they generate 100 terrabytes of data per day. This would be very high DATA.

CPLX:

Product Complexity: This metric is broken down into five categories: control operations which refer to the level of the nesting in the code, whether it is recursive or not and whether it is real-time or not, Computational Operations which refers to the level of complexity of the calculations made in the code, Device Depending Operations that deal with the hardware level work involved in a product, Data Management Operations which deal with the complexity of data structures, and finally UI management operations that deal with the complexity of the user interface graphically.

Story: Management has somehow secured a paying contract to make a terrible website that has only regular old HTML tags in it with no database, and no images. Sort of like a website from the late 1990's. This would be a very low CPLX.

RUSE:

Developed for Reusability: This metric describes the requirements level that a project's code be generic enough to use again for another project.

Story: Management determines that it is wasting money writing the same software over and over again after running and analysis and determining that similar code is being used in every project the company builds. They demand that the code from this project should be able to be reused on every project from here on out, no matter how different the application. This is Extra High RUSE.

DOCU:

Documentation Match to Life-Cycle needs: This is a metric for determining whether the documentation needs of a project over its entire life cycle are being met. The idea is that it's bad if a project is going to need a ton of maintenance but there's very little documentation.

Story: Management sells a long term maintenance contract to a client, but also commits to a very short development time frame. They decide that documentation should be the lowest priority task for all programmers in the company. This is very low DOCU.

TIME:

Execution Time Constraint: This metric concerns the amount of execution time available to be used by the project. To put it another way, this is the maximum amount of time that the software has to execute in terms of how much time is already in use.

Story: Management wants the developers to come up with a decision making system for a Game AI actor, but 95% of the execution time in the system is already being used by the great 3D graphics in place. This would be a situation with an extra high TIME.

STOR:

Main Storage Constraint: This metric is represents the same concept as TIME, but with on disk storage. The more storage space that is being used by everything else, the higher the main storage constraint.

Story: Management has set a hard constraint that the project use no more than 30GB of a user's disk space. 29GB are already taken up with a large number of features, but management has sold a large feature that needs to fit in that last 1GB. This is an extra high STOR.

PVOL:

Platform Volatility: This refers to both hardware and software, and is the amount of changes going on over time in each. Large numbers of changes to hardware and software over short periods of time yeild high PVOL.

Story: Management decides initially that everyone in the office should work on chromebooks with low amounts of memory, but a week later decides that everyone needs high powered windows machines. Then, 2 weeks later after a large number of complaints decides that everyone needs macs. During the same timeframe the network infrastructure is replaced twice, because to start it was low quality, and then the first contractor to work on it makes it worse, so someone else is called in. This is a very high PVOL.

Uppsies:

ACAP:

Analyst Capability: This is really just how good the analysts on a project are at their jobs. The metrics are: analysis and design ability, efficiency, thoroughness, and communication/cooperation skills.

Story: Management has forced all of their current analysts to quit prior to a new project. They hire 3 new analysts but decide to pay them all at the very bottom of the range for analysts. None of the analysts speak or write effectively in English, and the rest of the team are all English speakers. Additionally all of the new analysts are really terrible at their jobs. This is very low ACAP.

PCAP:

Programmer Capability: This is actually a measure of programmer's skill in working together as a team. It utilizes the same metrics that ACAP uses.

Story: Management fires all of their programmers and decides they need a multicultural staff, they hire 10 new programmers and none of them actually speak the same language or even write in the same language. Obviously all of the metrics involved are at the bottom of the spectrum since they basically can't work together at all. This is very low PCAP.

PCON:

Personnel Continuity: This is a measure of turnover in an organization.

Story. Management fires everyone in the organization every two months, they feel that everyone on the team just gets lazy if they stick around for more than two months. This would be very low PCON.

APEX:

Applications Experience: This is a metric for how much experience a team has in developing a particular type of application.

Story: Management decides they want the organization to get into front end web development. Their development team has only ever done back end web development. This would be very low AEXP.

PEXP:

Platform Experience: This is a metric for a team's knowledge of better platforms for their process in terms of UI, database, networking, and middleware.

Story: Management decides that there is a need for a move to an Oracle database for their products. Their current staff has no experience with Oracle so they hire several new employees that have more than 8 years of experience. Before the hiring PEXP would have been very low, after hiring the PEXP would have been very high.

LTEX:

Language and Tool Experience: This metric combines how much experience a team has with a particular language, with the amount of experience that a team has using the tool set that goes along with that language. Some of the tools captured in the metric are: requirements and design representation and analysis, configuration management, documentation extraction, library management, etc.

Story: The team at a particular organization specializes in projects in C++, but management has been hearing for months about how fantastic python is. So, in their wisdom, decide that everyone in the company must swtich to using python. None of the programmers in the company have even heard of python. This would be very low LTEX.

TOOL:

Use of Software Tools: This is a measurement of the complexity and robustness of the tools that are being used by an organization.

Story: Management decides that they are spending way too much programmer time on configuration and use of software tools. They decide that all of the programmers in the organization need to pare down their tool usage to only emacs or vim and a command line debugger. No other tools allowed. They would be creating a very low TOOL environment.

SITE:

Multisite Development: This measurement ties together the geographical level of distribution of an organization and their ability to communicate.

Story: Management decides that their staff needs quiet contemplation to perform their tasks. So they create offices on the most abandoned and connectionless places on the planet. Think an office on Easter Island or in the Galapagos Islands. The only problem is that they didn't budget for any communication, so no satellite phones, no internet, no real communication of any kind. This would be very low SITE.

SCED:

Required Development Schedule: This is a metric for how fast a product needs to be delivered. It measures the difference between the real schedule of a project and how much time a project that requires the same effort should take.

Story: Management decides that their staff is completely overworked, they have a new project coming up and know that their staff can do the project in 6 weeks, but they decide to bid 12 weeks so that everyone can catch their breath. This would be very high SCED.

Theory2: Treatments

doNothing()

-Story: Management decides that everything about their organization is perfect, so they don't actually do anything to change it. This is essentially no change to effort.

relaxSchedule()

-Story: Management decides that the schedule for a project is way too tight so increases the amount of time that the team will have to produce the product. But they actually increase the amount of time to 160% of what it actually should take for a project of this effort level.

reduceQuality()

-Story: Management decides that the projects they are bidding on are too quality dependent so they stop getting contracts with NASA for control systems and instead start getting contracts with local merchants for their websites. They don't provide any maintenance with their websites at all and the websites are all very simple websites with nothing more than HTML. This reduces the reliability requirement down to very low since none of these websites will actually have any sales elements to them, they are all just informational, very little documentation is needed since there's no maintenance contract, there's no execution constraint to speak of for the system since the websites are all static, and the complexity of the product is very low.

Theory3: Projects

-Differences between flight and ground:

- The xrange for kloc is (7, 418) in flight, and (11, 392) in ground

This means that they were less sure about how much code flight would take, as the range is slightly wider, and goes higher than the range for ground.

- CPLX for flight is [3,4,5,6] and for ground [1,2,3,4]

This means that they were pretty sure that flight was going to be a more complex program than ground, but in both cases were uncertain exactly how complex each would be.

-RELY for flight is [3,4,5] and for ground [1,2,3,4]

This means that the reliability required for flight was greater than the reliability required for ground, but the reliability required was uncertain.

-Differences between osp and osp2:

-SITE for osp is [3], and for osp2 is [6]

This means that in osp2 that there is better communication and employees are likely located in the same areas.

-FLEX for osp is [2,3,4,5] and for osp2 is [3]

For osp the organization was unsure of the level of flexibility they would have, but in osp2 they had hammered out their level of flexibility and it was near the middle.

-PREC for osp is [1,2] and osp2 is [3,4,5]

For osp the organization had never actually done software of this sort, but by the time osp2 came around the organization had much more experience in this area.

Theory4: Bad Smells

Stink[('sced','rely')] = Stink[('sced','pvol')]

Story: Management committed to a project schedule that was 75% of what that project should need. The project is flight control software for fighter jets, and every few weeks management decides that everyone in the team should work on a different operating system.

Stink[('pvol','pexp')]

Story: Management has hired a bunch of programmers that don't have much experience using linux, but all of their systems are linux based. They decide to switch to windows 2 weeks but then find out that none of the programmers have windows experience either. They continue this pattern, but never realize that the programmers don't have experience working on any platform!

Stink[('time','tool')]

Story: Management has decided the programmers on staff are not allowed to use any tools to build the new project they are working on, but the project has a very stiff requirement for execution time since the rest of the wider project is already using up 95% of the available execution time.

Practice

Task1: improvePersonnel increases the median bad smell value from 25 to 26, and decreases median effort from 1288 to 1278.

Task2: reduceQuality reduces bad smells but increases effort slightly. reduceQuality increases the median effort from 1288 to 1292, and decreases the bad smell median from 25 to 22.

Task3: improvesToolsTechniquesPlatform reduces bad smells from 25 to 22, and reduces effort from 1288 to 1279.