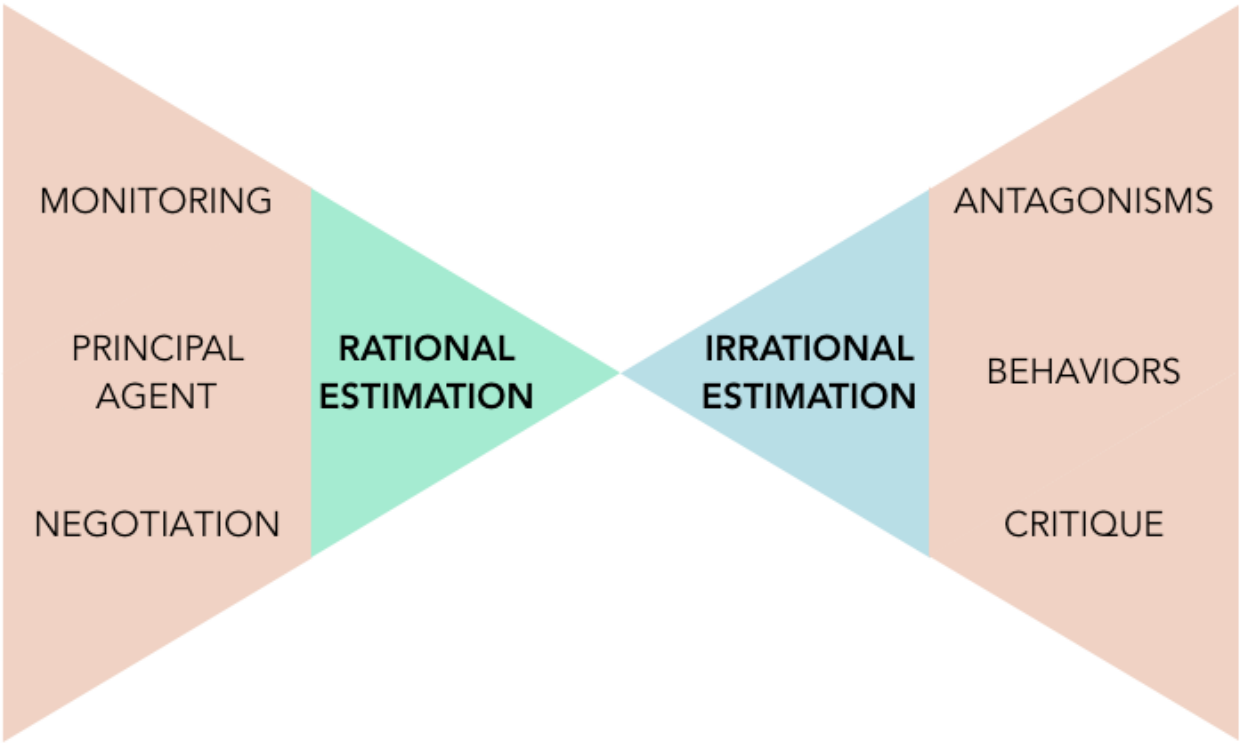
Self Organized Estimation

Commitment and Remorse

Is software estimation a fool's errand? If so, then how much more problematic would self organized software estimation be? Estimation within software has often been treated like a bike shed[1], so much so that the latest views seem to force developers to avoid hours altogether. This leaves us in the precarious positions such as that of converting agile points to hours for customers when they ask how long something will take. This seems to be a symptom of a deeper problem within software estimation. The central thesis here is that *negotiation position during the end game stage of development* is the determining factor of whether an estimation turns out to be valid or not.



Rational Estimation

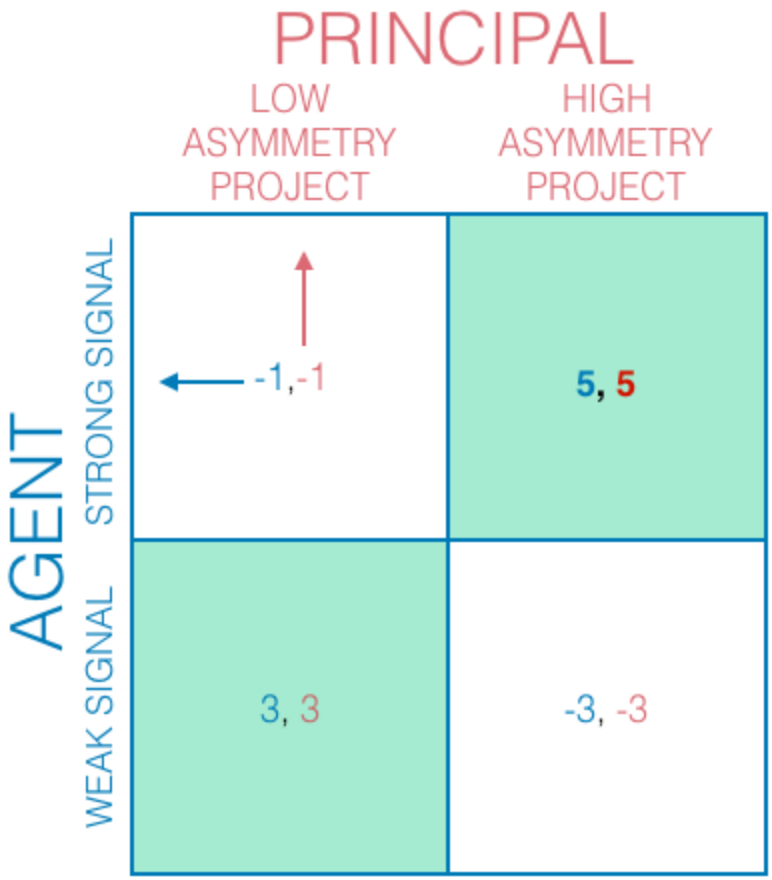
Contract theory is a form of game theory that applies to legal or bindable commitments between parties. The party that wants work to be done is called the principal, and the party that does the

[1] On Bike Sheds and Experts, Wavell Watson, https://s3-us-west-2.amazonaws.com/vulk-blog/OnBikeShedsandExperts.pdf

Copyright © 2016 W. Watson, Vulk LLC All Rights Reserved
http://www.vulk.coop

work is called the agent. Contract theory is useful to model agreements such as proposals which include estimations.

Within game theory there is a known issue called asymmetrical information. It occurs when one party has more information than the other party concerning important parts of the engagement. Within software development, one of the many ways asymmetrical information can occur is in the amount of work realistically required (versus contractually required), by either the principal (manager) or the agent (developer). The principal can increase the amount of work required through scope creep and the developer can increase the amount of work through over-engineering. Even in cases where the contract is a fixed bid, there is still the problem of incomplete contracts (in many cases it is impossible or cost prohibitive to completely specify the deliverables and contingencies in a contract).



In the game above, we will assume that the manager can either provide a low risk, well understood project (low asymmetry) or a high risk project with numerous unknown attributes (high asymmetry). The developer can either have a great reputation (a strong signal) or an unknown reputation (a weak signal).

Copyright © 2016 W. Watson, Vulk LLC All Rights Reserved
http://www.vulk.coop

Here are some stories that may help illustrate each engagement:

Low Asymmetry Project, Strong Signal Developer

During the performance of the project the manager will assume she knows how much effort you are giving, and will manage the project as such. She might not appreciate any creative ways you have for solving the problem and may micromanage a little too much for your tastes. After the project is complete the manager will be fairly certain about whether you did a good job or not. Even though you may have done a good job your manager may not be satisfied with your expensive cost since you essentially overqualified for the project. Likewise you might not be satisfied with the small amount that she did pay you. Your good name may be in jeopardy since your manager thinks you performed work that didn't seem worth the cost.

Low Asymmetry Project, Weak Signal Developer

You offer to do the work for a low price and this is exactly what your manager is looking for. There are better people out there at doing the job but she just wants something done quick and dirty. She doesn't even want to spend a lot of time looking for someone to do the job since it is simple in her eyes, so price is the most important thing. She might even be able to do the job herself but she just wants to delegate it. She can give you direction if you go too far off track anyways.

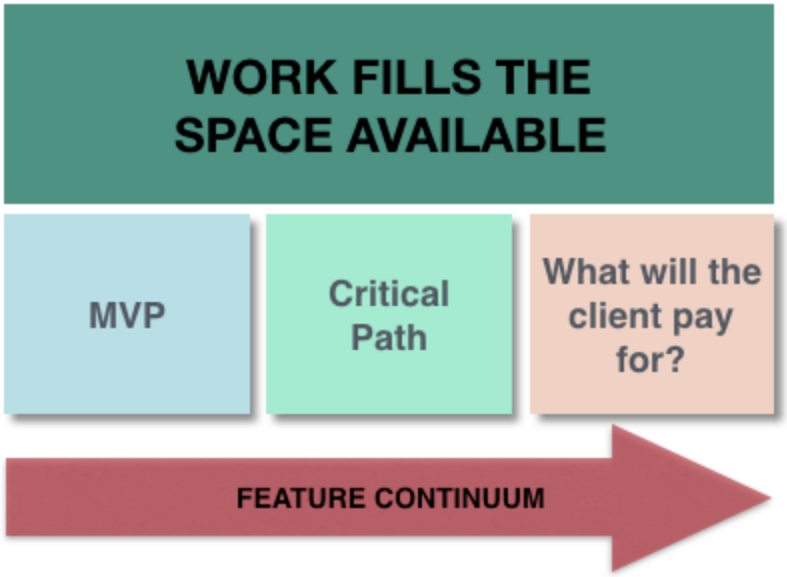
High Asymmetry Project, Strong Signal Developer

You have the ability to do the project and the credentials to go with it. Your project manager is bewildered by your creative solutions. You charge a high price and she knows that if she wants the best she has to pay for the best. She feels relieved that she isn't the one who actually has have to solve this problem but is still willing to have someone attempt to solve it even though there are no guarantees. If the project gets completed it will be worth the risk.

High Asymmetry Project, Weak Signal Developer

Even though your manager can't tell if you did a good job or not she will feel uncomfortable with your work since she can't measure your level of effort directly. She doesn't have a good feeling about you anyway because of your weak signal. This is true regardless of whether the project succeeds, because the unknown portions of your work may be poor.

Copyright © 2016 W. Watson, Vulk LLC All Rights Reserved
http://www.vulk.coop



The one thing to note about our narratives above is that whenever there is a strong signal developer or a high risk (high asymmetry) project, there is a risk of unneeded (inefficient) work being done. This can be looked at as Parkinson's law[2] applied to software, using a game theory model. In the case of a high asymmetry project, this work can't be seen. Economic moral hazard occurs when one party (usually represented as the agent) takes more risks because the other party (usually represented as the principal) bears the costs of those risks[3], such as the case in the high asymmetry project. The principal can also force risk onto the agent through scope creep and then withholding payment.

Monitoring ~ Asymmetrical information

Having a technical manager or liaison on the side of the principal can help with over-engineering if the project is trivial enough. Having a system change request process can help with scope creep, if the system change request process is *sufficiently painful* so as to deter use.

Paradox of the Analysis

The initial negotiation of an estimate, as far as rational negotiations go, can be described as a two-stage game wherein the first stage the principal facilitates some kind of contract for the work needed and then the agent accepts the contract. During the development of the contract or agreement there is at least one stage followed by one or more, usually hidden, stages. The initial stage will be some kind of rough estimate which will require some kind of analysis. The additional stages may be expected by the principal to be complementary, or free of charge. The paradox lies in that *the analysis for the estimation may itself need estimation*. If the analysis is

[2] https://en.wikipedia.org/wiki/Parkinson%27s_law
[3] https://en.wikipedia.org/wiki/Moral_hazard

Copyright © 2016 W. Watson, Vulk LLC All Rights Reserved
http://www.vulk.coop

small enough, perhaps a few hours, it may be able to be done for free. Often times this is found out to be not the case and the scope creep issue bleeds right into the free analysis.

If the principal is risk tolerant, the agreement might be time and materials (wherein the principal pays for the time and materials that the agent accrues). This allows the initial estimate to be rough or non-detailed. This also puts the risk exposure on the principal, since they will continue to pay until they run out of resources, even if the project is incomplete. If the principal is risk averse the agreement might be a fixed bid, which will put the exposure on the developer. Contracts for *custom* software that are fixed bid contracts are detailed. Fixed bids usually have a fudge factor of three or more times what the project is estimated to cost internally in order to cover that risk. Within the government space, the cost for doing any large custom software proposal (the vast majority of which are time and materials) is around 10% of the amount rewarded. That is to say that if the project is estimated to cost $10 million (a small project by federal standards) the *proposal* will most likely cost $1 million to create, given all of the analysis, marketing, and prep work involved.

So what is it that allows for a project to have rough estimates in a time and material agreement? The market is what decides this. If there are a lot of developers on the market who will do the detailed analysis required for a detailed estimate, then the principal will expect a detailed estimate. The negotiation about estimates included in the proposal begins at this point.

Irrational Estimation

Rational estimation assumes that the principal and agent both choose outcomes compatible with their own preferences. Irrational estimation is when a principal or agent *choose outcomes that are against what they consciously prefer*.

Antagonisms

An antagonism is a paradox or contradiction that can cause cognitive dissonance, fantasy screens, or symptoms in its adherent. *Estimation has an antagonism located within it that is the cause of irrational behaviors in estimators.* In order to understand this, we must understand the concept of a *limit* with respect to the *notion* of an estimate. At first glance, the definition of an estimate is a time and resource quantity that is given by an agent. A more developed definition, one that describes the intrinsic contradiction within an estimate, is that an estimate is a time and resource quantity *that accounts for the behavior triggered by the estimate.* That is to say that an estimate merely approaches true as it accommodates behavior. Why is this important? If we remember that work fills the amount of time given (either by feature exploration by the principal or architecture exploration by the developer), then the estimate must account for the fact that people will fill up the empty space left open by the estimate, perhaps in a hysterical manner,

Copyright © 2016 W. Watson, Vulk LLC All Rights Reserved
http://www.vulk.coop

while they search for the ‘right’ solution. That is to say that as the deadline approaches, the estimation is *retroactively* reinterpreted as always having been too aggressive.

Behaviors

There are numerous irrational behaviors that can appear in response to the estimation, that try to cover up the contradiction hidden in the notion of estimation.

Obsessional behaviors are when excessive rules are created or overbooking of resources occurs in order to forestall the deadline. An example of overbooking would be stakeholders scheduling themselves in less important meetings than the development meetings *even though it is clearly in their best interest to attend.* Excessive rules demonstrate obsessive behavior in the following manner: micromanagement of the agent’s behavior increases as the deadline approaches, which delays progress. In both cases the principal or agent *enjoys being careful.* This exceeds any risk averse description of a rational player to the point of becoming irrational and detrimental.

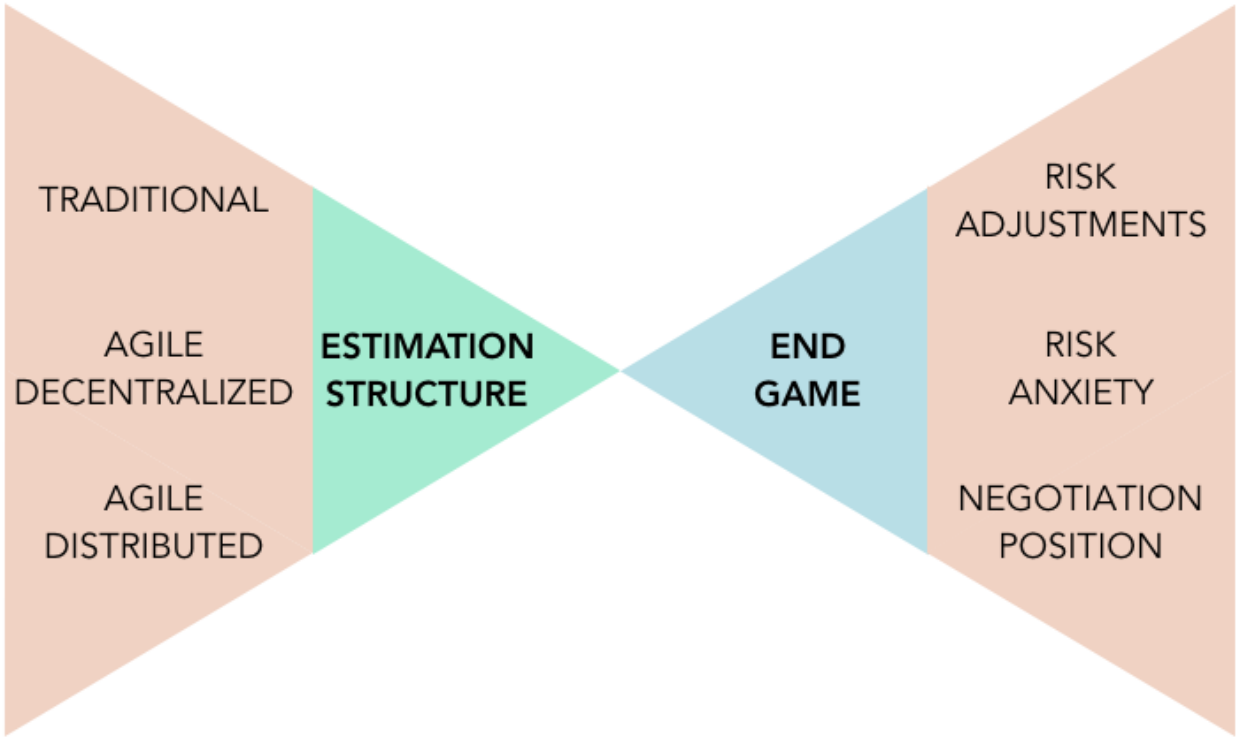
Hysterical behaviors are when deadlines are overshot because what is being created is *not ‘just right’*. This is the cause of scope creep, over-engineering, and over-designing. The effect is still *work filling the space available* but it occurs at the earlier checkpoints (e.g. sprints). An example of this is a UX expert desiring to get the typography *just right,* architects trying to obtain a certain standard in elegance of code that already works, customers adding yet another feature to the top of the priority list. As the deadline approaches the urge to *tweak*, *refactor,* and *add functionality* is perceived as part of the original estimate, even though the behavior fills out the gap left by the estimate. In hysterical behavior, the principal or agent *enjoys exploring.*

Perverted behaviors occur when a principal or agent succumbs to bureaucracy. This person *enjoys being able to navigate elaborate rules* and being part of that process. They also enjoy making others succumb to that process. Large companies and government agencies create processes (e.g. the federal procurement process) that can only be described as painful. There are people who enjoy being able to navigate this process (masochists) and administer the process (sadists). With respect to estimates, this type of principal or agent enjoys forcing the original estimate to succumb to the process.

Critiques

A critique is a subversive presentation or statement that forces us to confront an underlying paradox. Good design, analysis, and estimation processes will illustrate the tradeoffs and changes made to the original estimate, illustrating the ongoing negotiation or exploration so as to keep if from being hidden.

Copyright © 2016 W. Watson, Vulk LLC All Rights Reserved
http://www.vulk.coop



End Game

End game is the point during a competition (or cooperation) where the end of the engagement is known and strategic opportunities must be captured or lost. In their book Co-Opetition, Nalebuff and Brandenburger[4] illustrate end game using a card game in which there are two decks of cards, one black deck and one red deck. One student (Adam) has the black deck of cards and twenty six other students have one red card. If a player hands in a pair of cards (one black and one red) they get $100. No players may get together and collectively bargain; they may bargain only with Adam. There is also a set time to turn in the cards.

They go on to illustrate that at first glance it seems that Adam can end up offering you a small price, say $20, for your red card and keep $80 for himself leaving you with little room to negotiate. If you try to negotiate with Adam he can just stop negotiating with you and move on to someone else who will accept his terms. If you wait until all other players have already exchanged their cards with Adam, his only option becomes to trade with you for the last card or lose $100. At that point you can require a 50/50 split. Whoever else waits until the time requirement approaches will also be able to negotiate from the same position.

As the time constraint for an estimate approaches the *end game* emerges as an opportunity for a change in strategy. This is when principals may withhold payment, or agents may pause work. These are rational moves that are covered by contract theory as mentioned earlier. End

[4] Brandenburger, Adam M.; Nalebuff, Barry J. (2011-07-13). Co-Opetition (p. 42). The Crown Publishing Group. Kindle Edition.

Copyright © 2016 W. Watson, Vulk LLC All Rights Reserved
http://www.vulk.coop

game is also the point of anxiety in which irrational behaviors try to repress the approaching deadline.

Risk - Adjustments

The original estimate needs to be self-reflective in that it takes this end game period into account. In other words needs to take into account the risk that the other party will adjust to the estimate itself, in order to create a better negotiation position. Rational players always have the incentive to become hard negotiators during end game. Irrational players always have the highest amount of anxiety during the end game. The solution seems to be to set up a concrete negotiation position through the life of the project. The problem of imperfect contracts and short memories (irrational repression) presents itself here.



Historical Progression of Estimation

Copyright © 2016 W. Watson, Vulk LLC All Rights Reserved
http://www.vulk.coop

The numerous problems previously discussed have driven the progression of estimation from the traditional into the agile/decentralized project management eras. Large software projects have yet to fully adopt agile estimation procedures, but the amount of large projects using agile style estimations are increasing.

Traditional estimation

Traditional estimation tries to solve the problem aforementioned by doing big design up front and waterfall methods. The largest software projects (e.g. federal projects) are still planned, budgeted, and executed using some kind of big design up front proposals. The principal here will judge the agent based on past performance with other customers. For example, large procurement processes mandate three customers with similar projects as references in which the principal will ask questions about cost overruns and the like. The principal relies on the agent's signal before work is done to measure how useful the given estimate is.

Agile - Decentralized

Agile estimation uses the ready, fire, aim approach of measuring the velocity of a team in small iterations after it is already in progress. Each iteration of the team is measured, perhaps using points instead of hours, and recorded as the team's velocity. The team can then approach a customer with a measurement (e.g. twenty points per week) and then creates an estimate based on those measurements. Estimates are done by the implementers in the group. Each iteration is time boxed, low risk, and has it's own end game complete with deliverables and negotiation. These mini end games can get both sides familiar with *some* of the rational and irrational behavior involved with the party. There will still be an end game at the end of the project, however*, an experienced negotiator will reserve hard negotiation for the true end game.*

Agile - Distributed

Distributed estimation is virtually non-existent but the idea developed here is one of many ways that it could be done. The principal makes known a rough idea of the project that is needed to be developed as an RFP. The response to the RFP is a reverse bounty system where the agent proposes the smallest valuable deliverable with an estimate on it. The deliverable must be as close to a complete contract (the quality of service, performance, and delivery platform is described in the bounty) as possible. If the proposed service is accepted by the principal, the reverse bounty acts as a fixed bid with a time constraint. The time constraint can be extended, but if violated the RFP is re-issued and the existing contract is marked void. The pay is awarded automatically by verifying the deliverable via a Bitcoin Smart Contract. This form of procurement would lend itself to microservice development. The risk for both parties would be reduced because of an *executable contract* with no end game period. The risk involved with integration to other reverse bounties would be reduced because of the public nature of the contract. In the sense that the contract is a fixed bid with no chance of revocation, *it isn't an estimate*.

Copyright © 2016 W. Watson, Vulk LLC All Rights Reserved
http://www.vulk.coop

Conclusion

We participate in disavowal when we say ‘I know my act of estimating will be adjusted to, but nevertheless my estimate is ...’. We consciously know the act estimation is contingent upon forces that subvert the estimate, but we act like as this is not so. Given this, most attempts to estimate software are indeed a fool’s errand. Put in another way, a quality normally thought to have nothing to do with estimation, negotiation position, is possibly the most defining quality in whether estimates are valid. In order to bring this under control, we need subversive critique built into our development process that makes the exploration and negotiation effort painfully explicit. When the end game presents itself, we must be in a position to negotiate that the original estimate was accurate, or that it can be made to be accurate. Another solution here may be to sandbag[5] along the lines of the critical paths, and then present that overbuilt development during the end game, perhaps as complementary work, thereby reserving the end of the project to truly be for the ‘bells and whistles’. This would only work if ‘what they want’ could be determined from ‘what they asked for’ and then worked on separately.

[5] https://en.wikipedia.org/wiki/Sandbagging

Special thanks to the Vulk team

Copyright © 2016 W. Watson, Vulk LLC All Rights Reserved
http://www.vulk.coop