Agile product management using Effect Maps

Gojko Adzic
http://gojko.net

Effect Mapping is a game-changing technique for high level project visualization. It provides stakeholders and sponsors with an excellent level of visibility and helps to drive software projects towards delivering the right product with a high level of quality.

Effect Mapping facilitates the implementation of several techniques of agile planning, product design, prioritisation and scoping. In practice, I've found the combination of these techniques by far the most powerful way of iterative product management so far.

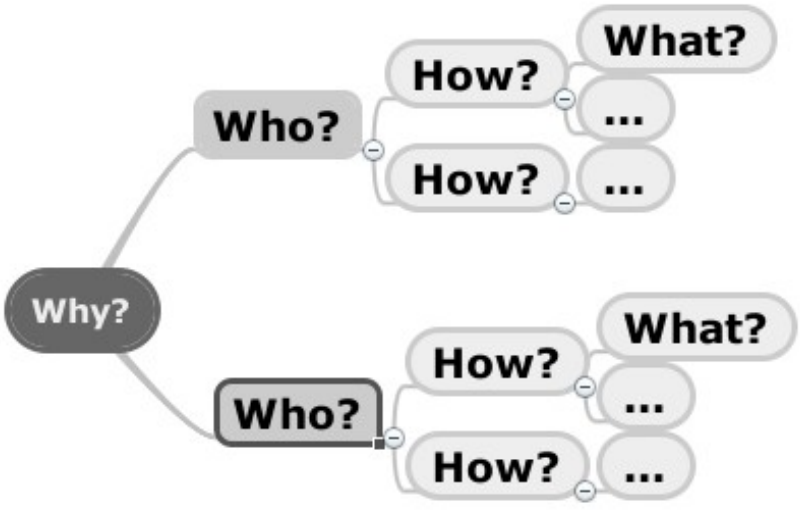
For the latest version of this article see http://gojko.net/effect-map

Introducing effect maps

Effect maps are charts of project scope which help teams ensure that software delivery is focused on business goals, stakeholders and their needs. Mijo Balic and Ingrid Ottersten present the technique in in *Effect Managing IT [Balic07]*, where they show that creating such a map helps to ensure that a project is focused on *achieving the desired business effect* (hence the name Effect Map). To create an Effect Map, draw a mind-map by answering these questions:

- *Why* are we doing this? What is the desired business change? This is the *business goal*. Put that goal in the centre of the map so that you can always keep that in mind.
- *Who* are the people that can create the desired effect? Who can contribute to the goal or affect it? These are the project *stakeholders.* Put the stakeholders on the second level of the mind-map.
- For each element on the second level: *How* can the target group contribute or obstruct the desired effect? In real life, not in software. These are *stakeholder needs*. Put them on the third level of the map.
- For selected elements on the third level: *What* are the business activities or software capabilities that would support the needs of the stakeholders? These are *features.* Features are at the fourth level of the map.

Figure 1. Effect Map Structure



In practice, I've found that asking sponsors for examples of how someone could help them achieve the goal is a great way to identify the stakeholders and their needs, in effect to drive the second and the third level of the map. Answers such as “For example, existing customers might help us by buying from us again” directly map to the *Who* and *How* levels.

Note that Balic and Ottersten ask *What* the target group wants on the third level and *How* a product should be designed on the fourth level. I found it more useful to ask the same questions differently. Asking *How* is a good way to tease out an example, so it is more useful to ask that when looking for stakeholders and their needs. Likewise, *What* is better when describing software features as it helps us focus on business functionality rather than implementation detail (I write more about this in [Adzic11]). As a consequence, the summary diagram in Figure 1 shows *How* and *What* on different levels than the summary diagram in Effect Managing IT, but the levels in diagrams are essentially the same from a semantic perspective.

I also allow non-software items to be on the fourth level, because software is not always the answer. Paying advertisers has nothing to do with software but might be a very effective way to contribute to a business goal and allow a delivery team to focus on building parts of software that have to be built.

The maps described in Effect Managing IT have only four levels. The teams I work with came up with ideas for the fourth level that were too big to fit into a single iteration, even too big to be user stories. I found it more useful to break down large feature ideas into several items that could contribute to that feature. In typical agile jargon, items on the fourth level could be epics. We can split them further into items on the fifth level or even sixth level, that represent user stories which can fit into an iteration. Another way to look at further levels could be feature themes at level 4, minimum marketable features at level 5 and user stories at level 6.

I also found it useful not to dive in too deep early. Going up to the level of stakeholder needs at the start proved to be quite enough for anything not immediately important. This approach allowed the teams I worked with to focus on the things that were important immediately but still keep a high level overview of the entire scope for later.

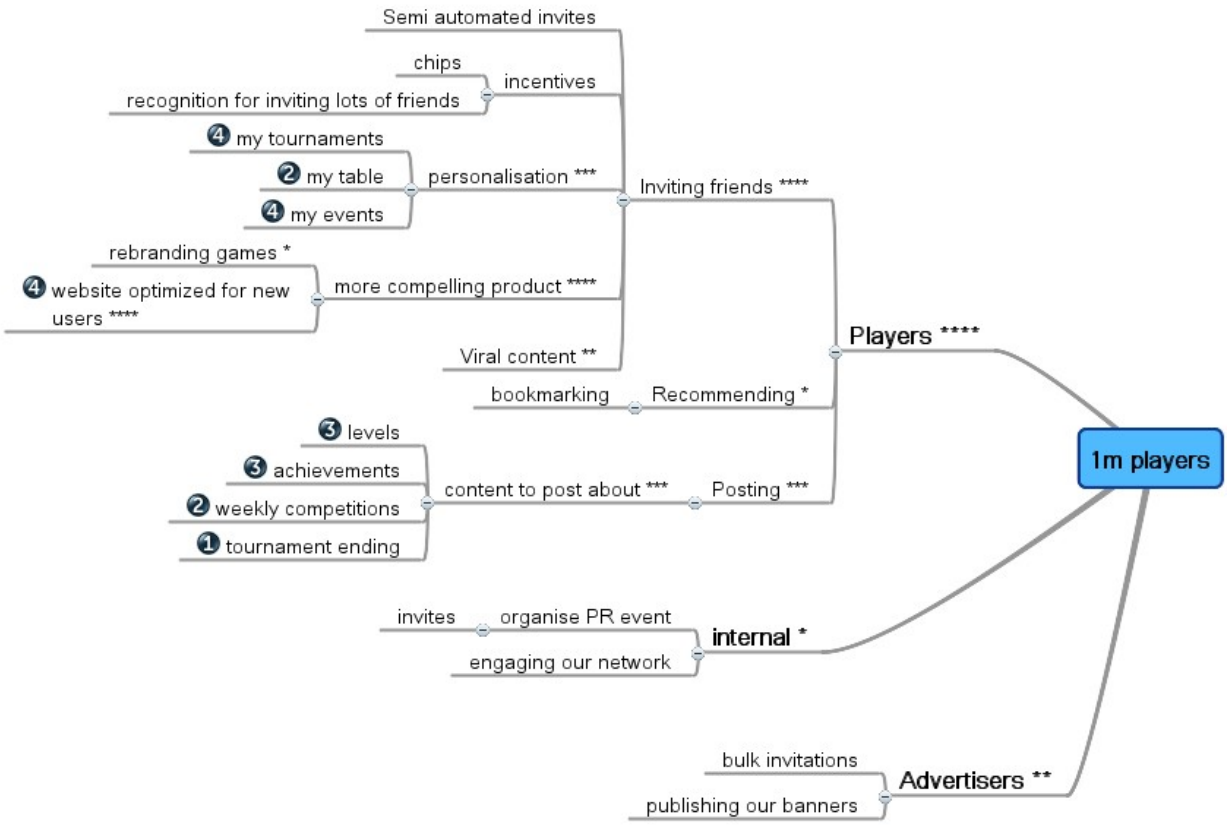
Not all stakeholder needs will be equally important or risky and not all software features will be equally complex. Visualising those aspects of needs, activities and features can help to identify low hanging fruit and to avoid death marches. Having this information on the map supports effective planning and prioritisation, so I extended the maps with simple visual symbols to represent importance and technical complexity of stakeholder needs and features. I use stars to represent importance and numbers to represent complexity. All these are rough, relative estimates, so don't worry too much about getting them right.

An example: Facebook games project

For an example, see Figure 2. This is a simplified Effect Map from one of my previous projects, an online games platform. The business stakeholders originally asked for levels and achievements in games as the next milestone. Instead of implementing that straight away, the team and the business stakeholders drew the map together.

It turned out that the reason why the business users asked for levels and achievements was to significantly increase the number of active players (the target was set measurably to one million). Once we have understood the goal, we started thinking about who could contribute to it. One obvious group were advertisers, who could send bulk invitations or publish our ads. Another group were the existing players, who could invite their friends, write about our games or recommend them to other people online. A third group was the development company itself, who could organise PR events or send out invitations. These high level examples directly translate into the second and the third level of the map.

Figure 2: An example effect map



We then added stars to mind-map elements to visualise the expected importance. Inviting friends works virally, because invited people can also invite others and it is a direct call to action. For that reason, it is likely to provide an exponential return on investment. Posting about games is not going to be that effective because it does not have a direct call to action. Recommending games is going to be even less effective because has a more limited reach.

We then came up with ideas how our software could support players in inviting their friends, posting about the games or recommending. We mostly focused on inviting and posting, as they seemed the most effective. With some branches we went to the fifth level, breaking down scope into smaller tasks and assigning numbers to those tasks to show a very rough estimate of implementation complexity.

The original requirements (levels and achievements), showed up on the map under giving existing players more content to post about. We decided together that inviting friends is more likely to bring large numbers because it is viral. So we focused first on redesigning the web site to provide incentives for invitations. Levels and achievements didn't come into the development pipeline for the next six months.

Product management using effect maps

Effect Maps are useful for much more than just an initial scoping exercise. I found them very useful as a catalyst in product management. They facilitate the implementation of several very good ideas for iterative product management, crucial for successful delivery of agile and lean projects:

- setting clear goals
- providing a shared understanding of quality
- prioritising based on business value
- iterative long term product release planning
- deriving scope from goals
- focusing deliverables on business value increments
- preventing scope creep
- supporting scope change and reprioritisation

Setting clear goals

If a project succeeds in delivering expected business value, it is a success from a business perspective. This is true even if delivered scope ends up being different than what was originally envisaged. On the other hand, if the project delivers exactly the requested scope to the word but misses the business target, it is a failure. This is true regardless of the fact that delivery teams can blame customers for not knowing what they want. Unfortunately, I've seen far too many projects where this business value is not clearly communicated to everyone. Very often it is not even defined, and in the cases when it is defined it is too vague to be useful. Such definitions make it hard to objectively measure whether the project actually delivered what was wanted. As a consequence, teams focus on ticking boxes by delivering scope as a measure of success.

Effect Mapping helps to define goals because it requires us to identify the expected goal as the first activity. But we can go further. Clear goals help teams design appropriate solutions. A solution to increase the number of players by 5% over 12 months is completely different than one that would increase the number of players by 100% over 6 months. To support the team in delivering the right system, we need to clearly define the business target. A great trick to ensure a shared understanding of the expected business effect is to decide how to measure it. After we have answered the question "*Why"*, we should go further and answer the following question:

- How will we measure how much any future delivery matches the expectations?

Defining the goal in a measurable way requires sponsors to think really hard and define precisely what they really expect out of a software project. That will help to align the expectations of the sponsors, the stakeholders and the delivery team. It will also allow the delivery team to design the appropriate solution and invest effort proportionally to the expected return.

Identifying measurable goals (such as "one million players" instead of just "more players") is key to ensure a shared understanding of what a project is supposed to deliver. Tom Gilb argues that measuring such effects and deciding how to measure them significantly improves the chances of success of a

project [Gilb05]. He also presents several techniques for measuring things that do not seem easily measurable. For further examples of measuring seemingly intangible things, see [Hubbard10].

Providing a shared understanding of quality

Similar to the way that project goals are often vague and not universally communicated, software quality is rarely defined precisely. This causes misunderstanding, misinterpretation and confusion about what a project needs to deliver. The interpretation of quality and the argumentation around that is delegated to a quality assurance role without much involvement from any business sponsors.

With an arbitrary interpretation of quality, there is nothing to help teams decide how much to invest in certain aspects of their product. Teams over-invest in less important parts of a system and under-invest in more important parts. To prevent this, we have to first clearly define what "appropriate quality" means and communicate that to everyone.

Although quality is often perceived as intangible, it isn't that hard to define. Gerald Weinberg defined quality as *value delivered to some person* [Weinberg91]. To specify quality, we have to identify the following two concepts:

- Who is that person? Or alternatively, who are the people affected by our work?
- What kind of value are they looking for from the system?

User Stories [Cohn04] apply a similar technique to ensure that each story delivers business value by requesting the writer to identify who is a stakeholder for a story and why they want it. In order to ensure successful delivery of milestones or entire projects, we need to define these aspects of our system not just on the low scope level (user stories), but also holistically for the entire project or a milestone of a project. In fact, I find that high level definition of quality much more important.

Effect Mapping facilitates this process because it requires us to clearly define the two aspects of quality – who the person is and what they expect – while drawing the map. They are effectively the second and the third levels of the map, the stakeholders and the stakeholder needs.

Prioritising based on business value

The hierarchical nature of the map clearly shows who benefits from a feature, why, and how that contributes to the end goal. This clear visualisation allows us to decide which activities best contribute to the end goal and where are the risks, which immensely helps with prioritisation. Once we have identified a clear goal, stakeholders and stakeholder needs, we can estimate how much we expect that supporting each one of them will contribute to the end goal. In the gaming system example, supporting invitations was clearly more important than supporting posting.

Effect Maps help us prioritise and invest appropriately in supporting activities depending on their expected value. In addition, this discussion provides a way to start thinking about how to measure whether a software feature has really delivered what we expect. Similar to the way how discussing how we can measure deliverables against a business goal, discussing how much a deliverable addresses a stakeholder need helps the team nail down what quality is and share the understanding.

Iterative product release planning

User stories are de facto standard today for managing long term release planning. This often includes an “iteration zero”, a scoping exercise or a user story writing workshop at the start of a milestone. During the “iteration zero” key project sponsors and delivery team together come up with an initial list of user stories that will be delivered. A major problem with the “iteration zero” approach is the long stack of stories that have to be managed as a result of it. Navigating through hundreds of stories isn't easy. When priorities change, it is hard to understand which of the hundreds of items on the backlog are affected. Jim Shore called this situation "user story hell" during his talk at Oredev 2010, citing a case of a client with 300 stories in an Excel spreadsheet. I've seen horror stories like that, perhaps far too often.

From my experience, project sponsors think about prioritisation in mid and long term in terms of the order of stakeholder needs they want to satisfy, not necessarily in terms of the order of system features. User stories try to address that but having too many stories upfront clutters the visibility, but having too few stories upfront doesn't give sponsors the confidence they need to support a project. Managing and delivering projects with dynamic scope scares people who are responsible for successful delivery. Many iterative planning and delivery practices might sound like black magic at a higher level and they do not allow managers to see the forest for the trees, so they push for huge lists of stories and ask for more control.

“People asking for control really want visibility”, said Elisabeth Hendrickson during her keynote at the Agile Testing Days 2010. She supported that statement with data from more than 100 workshops on team transitions she ran, which is as close to a scientific experiment as you can get with software development. The statement is certainly true from my experience. Clients and project sponsors want commitment and sign-off because they are afraid that their goals will not be met and because they have no visibility over software deliverables. User stories, as great as they are for short-term planning, are useless for high level visibility.

Effect Maps address this problem better by clearly visualising stakeholder needs and allowing us to prioritise at that level. Drawing an Effect Map up to the third level and sharing it ensures that stakeholders and their needs will be addressed. This also means that we only need the first three levels of the map for mid term and long term prioritisation and release planning, which enables us to postpone the discussion on detailed scope for everything but the highest priority stakeholder needs.

Defining the first three levels of a map in a measurable way ensures that there is a clear target for iterative delivery, without actually defining how we'll get there. Such definition of quality frees us to come up with the best possible scope, just in time when we need it, and not waste time on defining or managing scope too much up front. It also enables us to clearly provide visibility to project sponsors of how much we have delivered up to any point in time.

Effect Maps allow us to provide good visibility instead of control. As the project progresses, we can mark areas of the map that have been delivered, identify stakeholders whose needs are satisfied and plan whose needs will be fulfilled next. This high level visibility provides the delivery team and the project sponsors enough information to track progress and plan effectively further scope iteratively. In addition, this high level visibility enables people to see the big picture so that they can prioritise and analyse further scope for immediate delivery.

Deriving scope from goals

Techniques for collaboratively deriving scope from goals, such as *Feature Injection* [Matts09], are becoming increasingly popular in software development. Although still in an early adoption phase, such techniques are used by teams to build in quality from the start and further enhance the effects of techniques such as specification by example [Adzic11].

Effect Maps facilitate this process by helping a team to clearly focus on a business goal while planning scope. Drawing an Effect Map requires us to define implementation scope to satisfy a business goal. By allowing us to postpone the discussion on system features too soon, Effect Mapping enables us to derive scope from goals just in time. This fits in nicely with flow-based software delivery methods, such as Kanban.

Focusing deliverables on business value increments

Focusing delivery on increments of business value instead of increments of technical functionality provides fast feedback to project sponsors about their assumptions in iterative deliveries. It also supports the delivery team in learning more about the business domain and technology. This is a key element of planning with User Stories (see [Cohn04]). Focusing delivery on the most valuable Minimum Marketable Features (MMFs) allows the management to plan releases to achieve the best level of return on investment from the project [Denne03].

By focusing our planning and prioritising on addressing stakeholder needs, we ensure that deliverables actually provide business value. Effect Maps facilitate structuring a project around stakeholder needs. They help us break down how to satisfy stakeholder needs into MMFs and User Stories hierarchically.

In the “iteration zero” approach, teams need to define a large number of stories quickly. They focus on deliverables (“I want ...” part), and invent stakeholders (“as a ...”) and benefits (“in order to...”) half-hazardly to obey the form. Stories become generic such as “as a trader I want to trade so that I can trade”, with dozens or hundreds of stakeholders and benefits. This completely defeats the point of user stories that are supposed to communicate intent. The need to fit stories into something easily deliverable in isolation disconnects the deliverable piece from the value expected by the customer, in particular when the benefits and stakeholders are invented on the fly. Note that this is not an issue with user stories as a technique, because they were designed to deal with this very problem, but an issue with how teams commonly misuse them.

Effect Maps facilitate the correct application of User Stories as they help us directly answer who is a stakeholder for a particular feature and how that feature helps the stakeholder achieve something useful.

Preventing scope creep

With a clear map from features to the end goal, it is easy to spot if a particular suggested feature is not relevant for the ultimate goal of a project. With large flat backlogs of ideas it is less easy to spot such things. Effect Maps achieve the same effect as the Goals-Features-Requirements hierarchies described in [Berkun05], ensuring that everything in scope really contributes to the goals. This helps to avoid scope creep by providing an argument against unnecessary pet features.

Supporting scope change and reprioritisation

Business priorities really change at the level of stakeholder needs. If a product backlog is described with an Effect Map, the team can effectively react to such changes.

Effect Maps allow us to derive scope incrementally and just in time. We do not have to make many decisions on scope for individual stakeholder needs before we really need to start investing in them. Even if we have already defined several levels of features for the reprioritised stakeholder needs, there is a clear hierarchical map that shows us what gets moved. We can stop working on any features related to needs that do not need to be satisfied now and start working on more important business deliverables. With product backlogs that are just linear stack of stories, we don't have anywhere near this kind of visibility, which is one of the reasons for “User Story Hell”.

Shifting the mindset from cost of IT to investment in IT

When we discuss priorities and plan without an assumed software implementation scope, my assumption is that we can change the tone from how much is it going to cost/take to have something delivered to how much the sponsors want to invest in supporting a particular stakeholder need. (This is the only assumption in this paper, I've tried all the other ideas in practice successfully). By applying Gilb's measurability ideas to how much satisfying stakeholder needs will contribute to a goal, we can get an idea of what is the expected outcome of the related set of features, which will guide us in deciding how much to invest.

With defined expected outcomes and investment, a team can come up with the suitable set of software features and enhancements to support the investment and achieve the goal, or decide that such a delivery is unrealistic early on.

Instead of estimating how long a set of features will take to develop, we can invest in building features up to a certain level of quality. This is where User Stories fit in perfectly and where a clear definition of quality ensures a shared understanding. By delivering the features - fourth and fifth level deliverables - we should get parts of the expected results for parts of the investment. We can then measure the outcomes of increments and validate our business assumptions early.

For example, in order to increase the number of users (“why”**)**, a company might want to stimulate existing customers (“who”) to invite their Facebook friends to sign up for a product (“how”). The business can decide to invest 100.000 GBP and one month of development time in improving the way fans can invite their friends to sign up for a product through Facebook, expecting that this will give them 50.000 new users over 6 months. The team can then suggest ways for the software to support such an investment, for example offering some kind of personalisation (“what”), which can lead to 5-6 suggestions of user stories that improve the software in that area. Each user story should ideally deliver a slice of the value, so it should create a visible effect on the number of new users. Although the expected total improvement is specified on the level of six months, the sponsors and the delivery team can come up with a scaled down short term indicator of what they expected from a particular story.

After the first few stories go live, the sponsors can check their assumptions about the value of Facebook invitations against those indicators much before the entire investment is spent. This can either validate the assumptions and confirm that further investment in invitations or personalisation is justified, or lead to a reprioritisation from a business level because the indicators are not there. In

addition, if the expected outcome is achieved sooner, for example if the few early stories already deliver much more than the expected indicators, the business might decide to stop investing in invitations and shift the focus somewhere else.

Conclusion

Effect Mapping, as a process, helps to drive the project implementation scope towards delivering the right business effects. The resulting mind maps provide a good visualisation of software project scope for the sponsors to effectively track progress, plan and prioritise without micro-managing and getting involved in minute delivery tasks. Effect Maps facilitate the implementation of User Stories for short term planning, measurable well defined outcomes for business prioritisation, focusing on minimum marketable features for highest return on investment. They help teams to implement other good iterative product management practices. In addition, my assumption (which I plan to verify the first time I get the chance) is that it will help to switch the mentality from cost to investment for IT projects, and facilitate the business delivery for the return on that investment.

References

- [Balic07] Mijo Balic, Ingrid Ottersten, Effect Managing IT, 2007, Copenhagen Business School, 978-8763001762
- [Cohn04] Mike Cohn, User Stories Applied for Agile Software Development, 2004, Addison-Wesley Professional, ISBN 978-0321205681
- [Weinberg91] Gerald Weinberg, Quality Software Management, 1991, Dorset House Publishing, ISBN 978-0932633224
- [Adzic11] Gojko Adzic, Specification by Example, 2011, Manning, ISBN 9781617290084, http://specificationbyexample.com
- [Berkun05] Scott Berkun, Art of Project Management, 2005, O'Reilly, ISBN 978-0596007867
- [Denne03] Mark Denne, Jane Cleland-Huang, Software by numbers, 2003, Prentice Hall, ISBN 978-0131407282
- [Gilb05] Tom Gilb, Competitive Software Engineering, 2005, Butterworth-Heinemann, ISBN 978-0750665070
- [Hubbard10] Douglas W. Hubbard, How to Measure Anything, finding the value of “intangibles” in business. Wiley, ISBN 978-0-470-62568-2
- [Matts09] Chris Matts, Real Options, http://www.lulu.com/product/file-download/real-options-at-agile-2009/5949486

Gojko Adzic is a frequent speaker at leading software development and testing conferences and runs the UK agile testing user group. Over the last twelve years, he has worked as a developer, architect, technical director and consultant on projects delivering financial and energy trading, mobile positioning, e-commerce and online gaming.

Gojko runs Neuri Ltd, a UK based strategic consultancy that helps ambitious teams from web startups to large financial institutions improve software delivery.

To get in touch, write to gojko@neuri.co.uk or visit http://gojko.net.

For more information on Gojko's latest book, see http://specificationbyexample.com