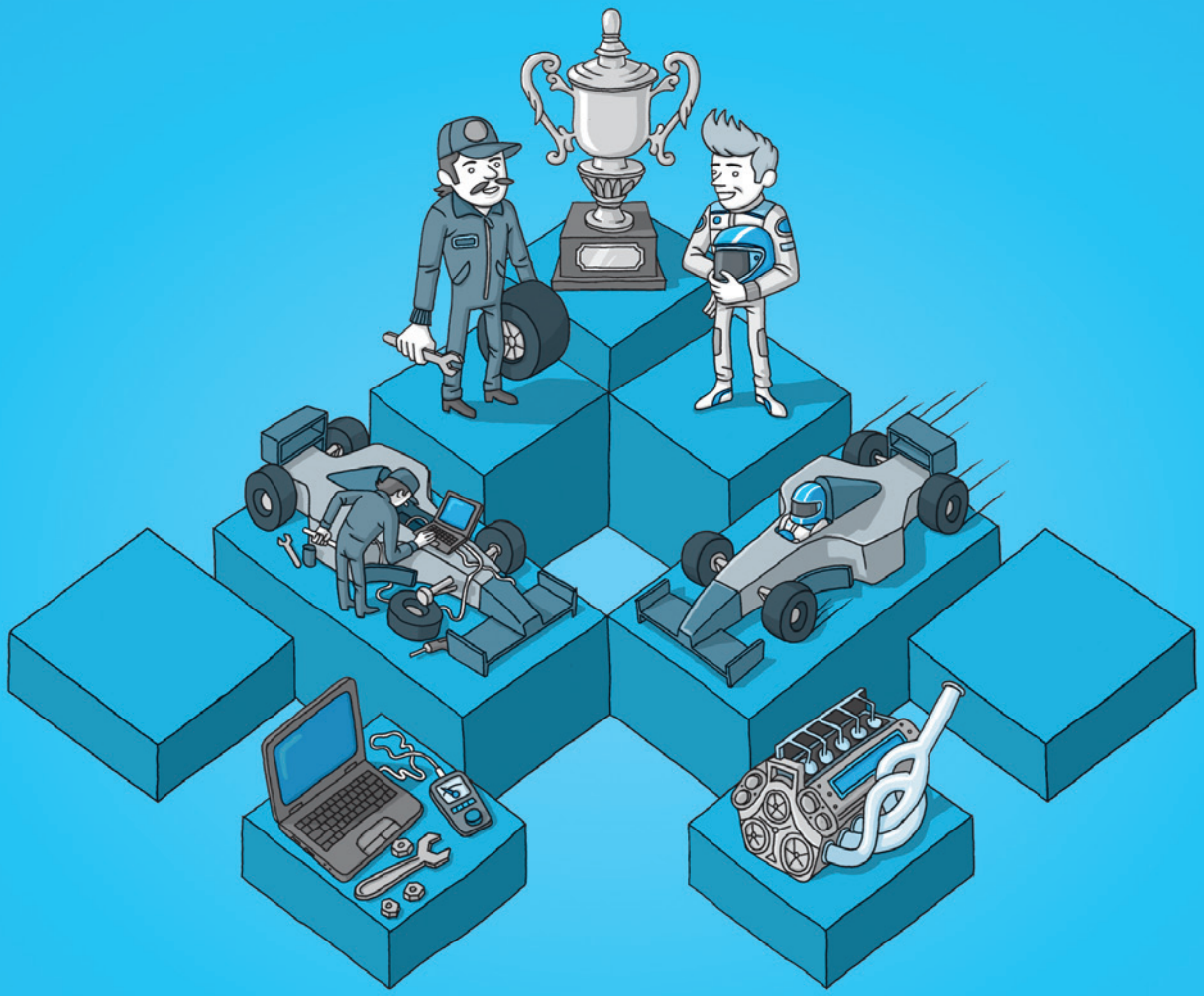
GOJKO ADZIC

IMPACT MAPPING

MAKING A BIG IMPACT WITH SOFTWARE PRODUCTS AND PROJECTS



foreword by **Tom Poppendieck**

Impact Mapping: Making a big impact with software products and projects

PRINT ISBN: 978-0-9556836-4-0

Published 1 October 2012

Author: Gojko Adzic
Copy-editor: Marjory Bisset
Design and layout: Nikola Korac

Provoking Thoughts Limited
Menzies Suite A, 1st Floor
62 Goldsworth Road
Woking, Surrey GU21 6LQ
United Kingdom

CONTENTS

WHAT IS AN IMPACT MAP?

THE ROLE OF IMPACT MAPS

CREATING AN IMPACT MAP

GIVING CREDIT WHERE CREDIT IS DUE

Impact mapping is a variant of the InUse effect mapping method, introduced by Mijo Balic and Ingrid Domingues (Ottersten), combined with impact maps for training organisations invented by Robert O. Brinkerhoff, the feature injection ideas of Chris Matts, the measurability and iterative delivery ideas of Tom Gilb. It draws heavily on their work – enough to say that all key good ideas are theirs, I've just linked them together and put them into the perspective of modern software delivery practices. You will find references to the original ideas at the end of this booklet. Most of the glue between these ideas comes from inspiring, insightful and challenging conversations I've had with Craig Larman, Tom and Mary Poppendieck, Dan North, Gordon Weir, Jeff Patton and Matthias Edinger (in no particular order).

By combining these ideas, impact mapping brings usability and speed to proven product and project management strategies, helping them fit better into modern software delivery constraints, and at the same time applying some great ideas from other industries to software delivery.

This book describes the way I use impact maps. In my previous work, I referred to the method described here as "effect mapping", as the structure closely resembles InUse effect maps. However, the way I use the maps is significantly different from the InUse method. I follow an approach much closer to what Brinkerhoff describes in his HET (highly-effective training) method, as roadmaps and iteratively refined milestone plans. In addition, I found that a slightly modified set of questions fits better the kinds of projects that I am involved in. InUse effect maps aim to facilitate innovative product design and user experience design. As a consultant, I work with ambitious organisations to help them improve delivery practices, and they often suffer from scope creep, lack of big picture, lack of alignment of delivery teams with business objectives; they waste a lot of time and effort building the wrong software. Impact mapping is a fantastic way to reduce all that suffering.

Using the same name for both InUse effect maps and the maps as I use them caused confusion, to the point that a well-known consultant said to one of my clients, "Gojko got effect maps completely wrong, but he's on to something". After several conference presentations in Sweden, the home of InUse, attendees came to complain that I was presenting effect mapping wrongly. This is why in this book, and from now on, I will use the name impact maps instead of effect maps for my method. By using a different name, I hope to prevent further confusion.

Craig Larman suggested the name "impact maps", which is similar enough to effect maps but different enough not to cause confusion. Brinkerhoff calls his planning visualisation impact maps, which justifies the use of that name. His maps are used for managing training plans for organisations, so I hope that there is not much danger of confusion.

Disconnecting the maps I describe here from InUse effect mapping by name also allows me to focus completely on managing scope for software delivery, and use names for map elements that are more appropriate for that context.

WHY SHOULD YOU CARE?

Impact mapping is a strategic planning technique. It prevents organisations from getting lost while building products and delivering projects, by clearly communicating assumptions, helping teams align their activities with overall business objectives and make better roadmap decisions.

Our products and projects do not work in a vacuum. They have an interdependent, dynamic relationship with people, other projects, the organisation and the wider community around them. Yet currently popular planning methods either expect the world to stand still while we deliver or give up on creating any kind of long-term big-picture view, leaving a huge communication gap between business sponsors and delivery teams. Impact maps visualise the dynamic relationship between delivery plans and the world around them, capturing the most important assumptions as well as delivery scope. They help us adapt plans effectively and react to change, while still providing a good road map for delivery teams and a big-picture view for business sponsors.

Impact mapping helps to reduce waste by preventing scope creep and over-engineered solutions. It provides focus for delivery by putting deliverables in the context of impacts they are supposed to achieve. It enhances collaboration by creating a shared big-picture view for technical and business people.

Similar approaches to impact mapping are used in many proven management styles, including US and UK military Mission Command and workplace empowerment in commercial organisations, as well as several goal-oriented requirements engineering methods for software. Impact mapping has several unique advantages over similar methods:

- It is based on a method invented by an interaction design agency and similar to a team-building method, which means that it facilitates collaboration and interaction. It is significantly less bureaucratic and much easier to apply than many alternatives. It also facilitates the participation of groups of people from different backgrounds, including technical delivery experts and business users, helping organisations use the wisdom of crowds.

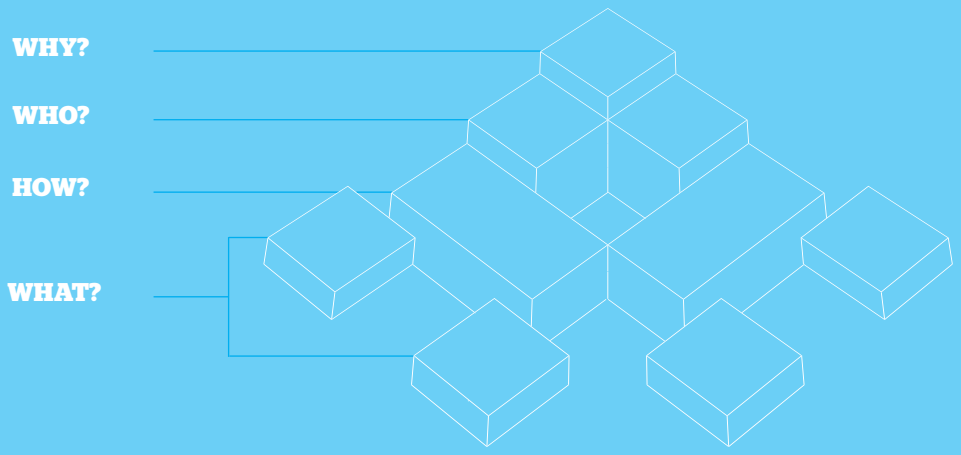
- It visualises assumptions. Alternative models mostly do not communicate assumptions clearly. Impact mapping does, and because of that it helps teams to make better decisions in rapidly changing environments such as IT. The visual nature of this method also facilitates effective meetings and supports big-picture thinking, which provides organisational alignment.

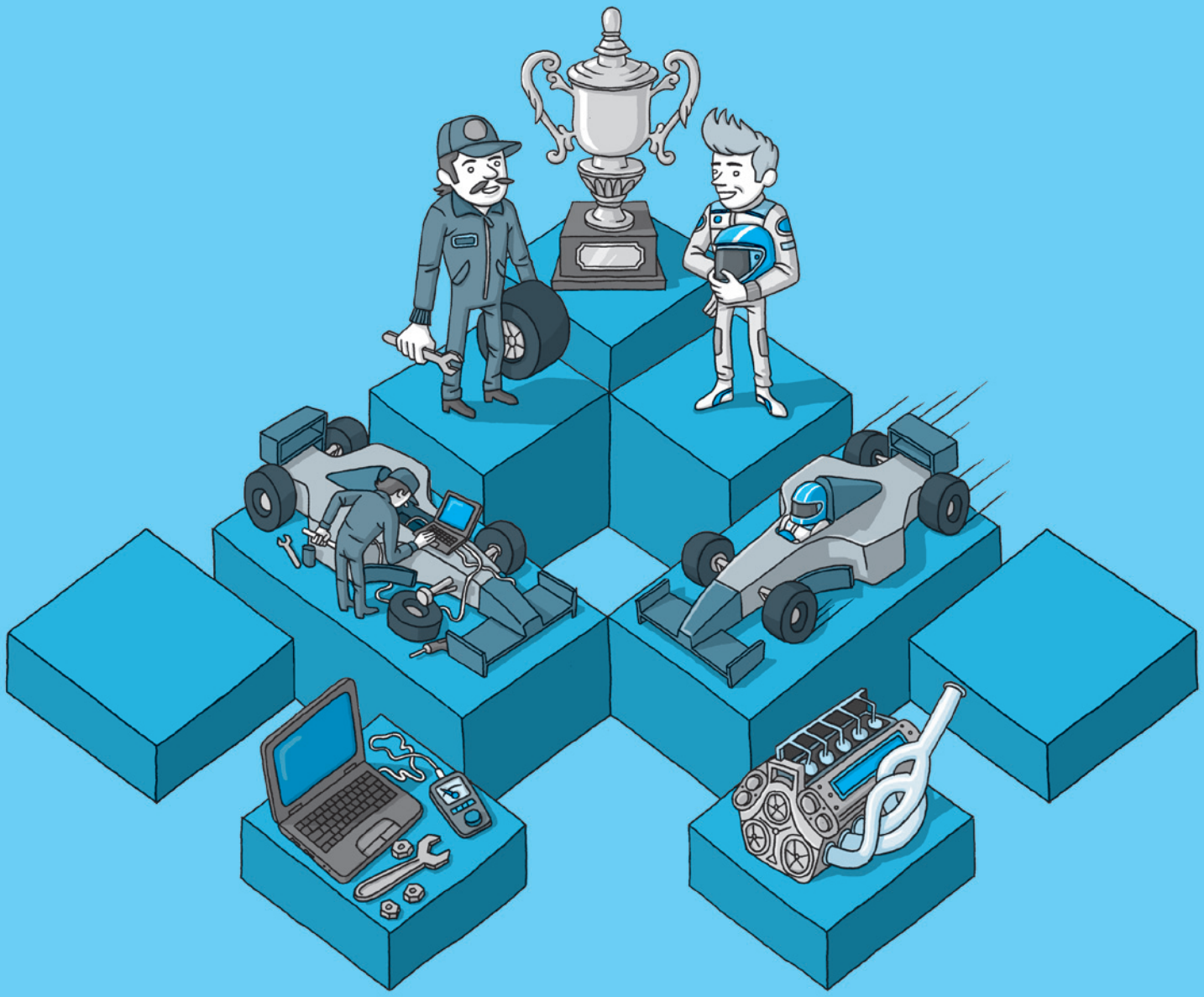
- It is fast. One of my clients recently said that it would take them months to achieve what we did in just two days. Because of this, it fits nicely with iterative delivery models that are now becoming mainstream in software.

In essence, you should care about impact mapping because it can help you build products and deliver projects that make an impact, not just ship software.

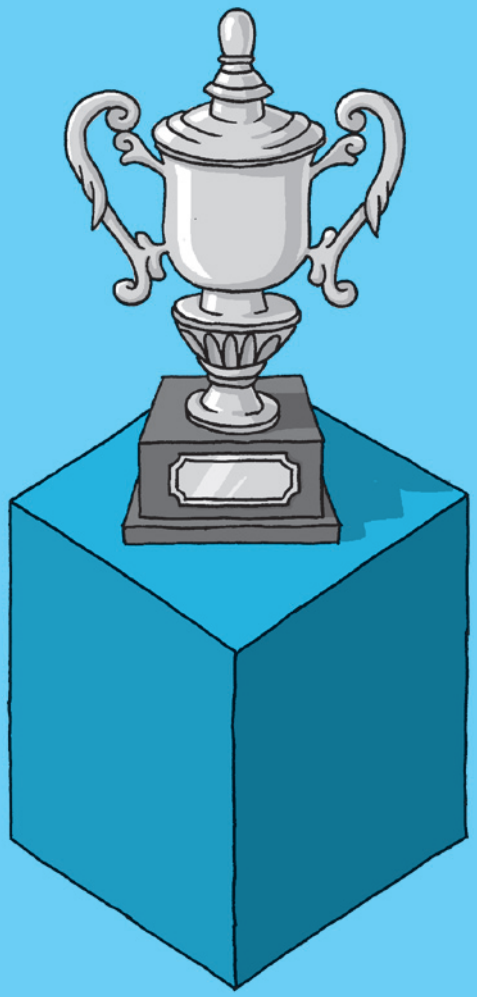
WHAT IS AN IMPACT MAP?

An impact map is a visualisation of scope and underlying assumptions, created collaboratively by senior technical and business people. It is a mind-map grown during a discussion facilitated by answering the following four questions:





WHY?



GOAL

The centre of an impact map answers the most important question: *Why are we doing this?* This is the **goal** we are trying to achieve.

It might sound like common sense to know this upfront, but from my experience very few people working on delivery know the actual expected business objectives. These are sometimes drafted in a vision document, but more frequently exist only at the back of senior stakeholders' minds. Even when they are communicated, business goals are often defined in vague terms. Knowing why we're doing something is the key to making good decisions about cost, scope and timelines, both at the start and later when things change.

Gary Klein's research with emergency services and military personnel shows that people on the ground have to know the objectives of any activity in order to react correctly to unforeseen problems. And unforeseen problems are a fact of life in any but the most trivial software. If a product milestone or project succeeds in delivering the expected business goal, it is a success from a business perspective, even if the delivered scope ends up being different from what was originally envisaged. On the other hand, if it delivers exactly the requested scope but misses the business goal, it is a failure. This is true regardless of the fact that delivery teams can blame customers for not knowing what they want.

By having the answer to 'WHY?' in the centre, impact maps ensure that everyone knows why they are doing something. That helps teams align their activities better, identify true requirements and design better solutions.

GETTING IT RIGHT

The purpose of a goal definition is to allow the delivery organisation and business sponsors to re-evaluate the plan as new information becomes available. For this reason, good goals tend to be SMART: Specific, Measurable, Action-oriented, Realistic and Timely.

Goals should not be about building products or delivering project scope. They should explain why such a thing would be useful.

Goals should present the problem to be solved, not the solution. Avoid design constraints in a goal definition.

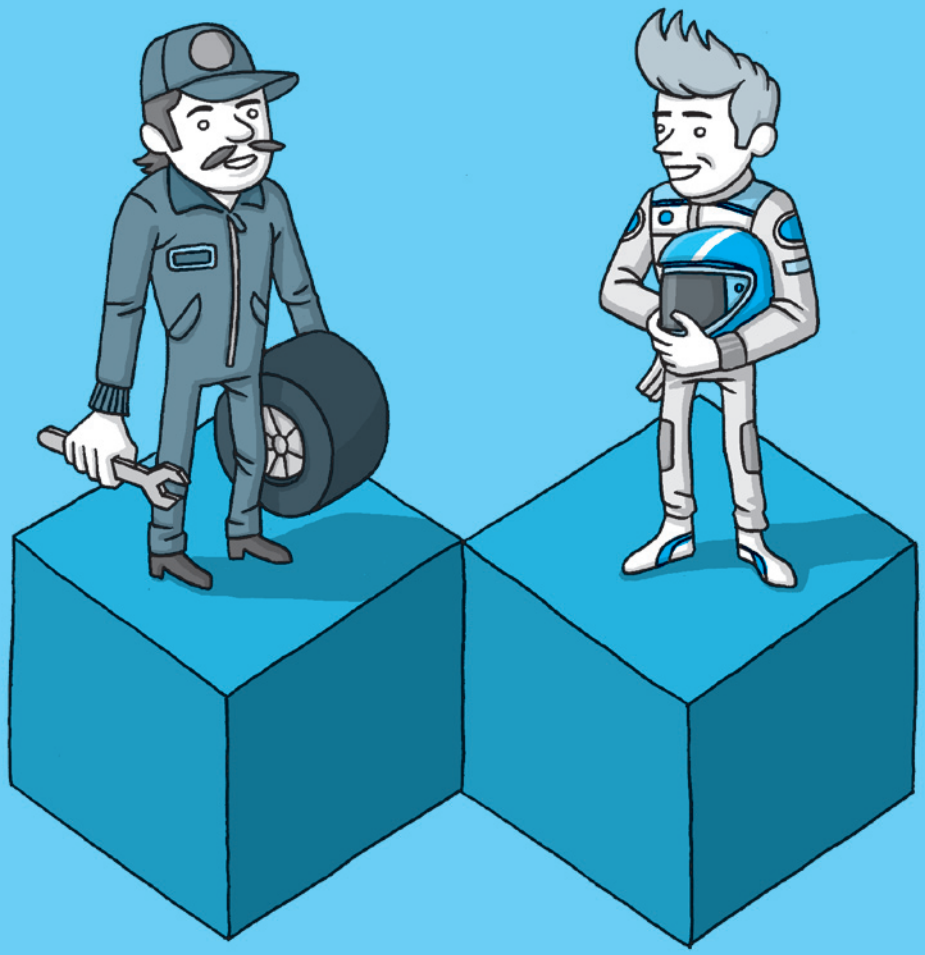
Don't worry about nailing it down to a single number. Chris Matts suggests defining a model for business value first, and then defining goals as increments of business value, explaining how the situation should change in the future. This is particularly effective when you have a set of key performance indicators for product performance.

For commercial products and organisations, try to define goals that have an obvious link to money.

EXAMPLES

- *Starting to trade in Brazil by March next year*
- *Increasing user conversion by 20% in three months*

WHO?



ACTORS

The first level of an impact map provides answers to the following questions: *Whose behaviour do we want to impact? Who can produce the desired effect? Who can obstruct it? Who are the consumers or users of our product? Who will be impacted by it?* These are the **actors** who can influence the outcome.

Gerald Weinberg defined quality as 'value delivered to some person'. To deliver high-quality results, we first have to understand who these people are, and what kind of value they are looking for from our products or project outcomes. In addition to those directly getting value out of our software, we also have to consider a host of others who can make decisions that influence the success of a product milestone or the outcome of a project. Software does not work in a vacuum and it rarely controls all the actors who are involved with it. People have their own needs, goals and preferences, which all come into play if we truly care about achieving a business goal instead of just delivering software.

Yet most requirements models completely ignore this – they focus on what the software should do and not who will benefit from it and who will be worse off when it is delivered. Then somewhere mid-work a new actor appears from nowhere and everything changes fundamentally, or someone with sufficient decision-making influence just stops the delivery in its tracks.

Impact maps make us think about all these decision-makers, user groups and customer segments. By mapping out different actors, we can prioritise work better – for example focusing on satisfying the most important actors first.

GETTING IT RIGHT

Important actors are those who can significantly influence the success of a project or product milestone, including end-users and internal or external decision-makers. Alistair Cockburn advises looking for three types of actors:

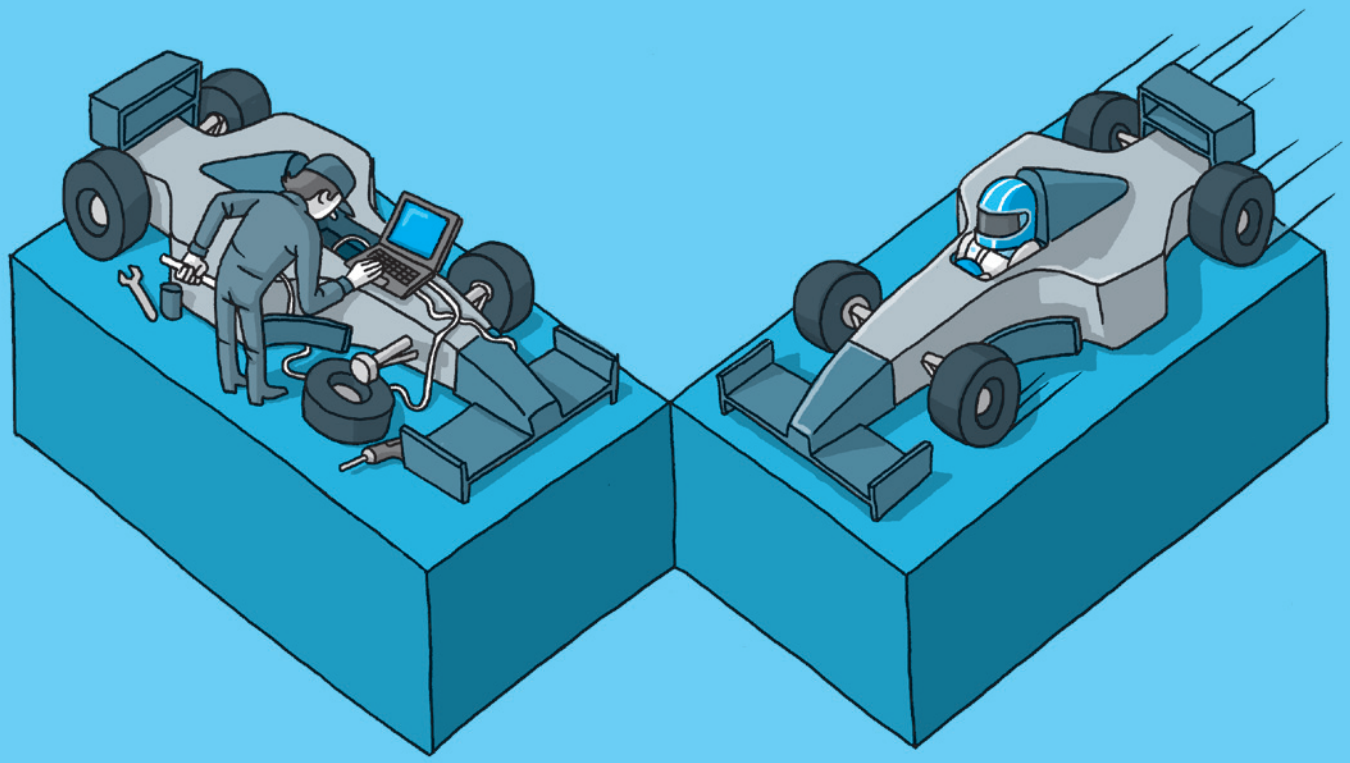
1. *Primary actors, whose goals are fulfilled, for example players of a gaming system*
2. *Secondary actors, who provide services, for example the fraud prevention team*
3. *Off-stage actors, who have an interest in the behaviours, but are not directly benefiting or providing a service, for example regulators or senior decision-makers*

Be specific. Avoid generic terms such as 'users' – different categories of users might have different needs, and not all users of a system might be important to consider for a particular project. Try to define actors in this order: specific individual, user persona, role or job title, group or department.

EXAMPLES

- *Mike Smith from the marketing department*
- *Under-18 users with a mobile device at a concert*
- *Apple ITunes store approvers*

HOW?



IMPACTS

The second level of an impact map sets the actors in the perspective of our business goal. It answers the following questions: *How should our actors' behaviour change? How can they help us to achieve the goal? How can they obstruct or prevent us from succeeding?* These are the **impacts** that we're trying to create.

Anthony Ulwick wrote that a key to successful delivery is to understand what jobs customers want to get done instead of their ideas about a product or service. This helps delivery organisations investigate different technical options and explore solutions to produce good results. It also helps to focus delivery on supporting users in getting the job done instead of just delivering features. Robert Brinkerhoff suggests something similar, but focusing on desired changes in jobs instead of just jobs that users are trying to achieve.

By listing impacts on the second level of a map, we consider the desired changes in the behaviour of actors. This leads to better plans and helps with prioritisation. Different actors could help us or obstruct us in many ways on our route to achieving the key business objectives. Some of the impacts will be competing, some conflicting, some complementary. We do not necessarily have to support all of them, but without considering delivery scope in the context of these activities it is very difficult to prioritise and compare deliverables. The hierarchical nature of the map clearly shows who creates an impact and how that contributes to the goal. This clear visualisation allows us to decide which impacts best contribute to the goal and identify the risks; this helps immensely with prioritisation.

GETTING IT RIGHT

Don't list everything an actor might want to achieve. List only the impacts that really help move you in the right direction towards the central goal.

Impacts are not product features. Avoid listing software ideas here, focus on business activities.

Ideally show a change in actor behaviour, not just the behaviour. Show how the activity is different from what is currently possible. So instead of just 'selling tickets', say 'selling tickets five times faster'.

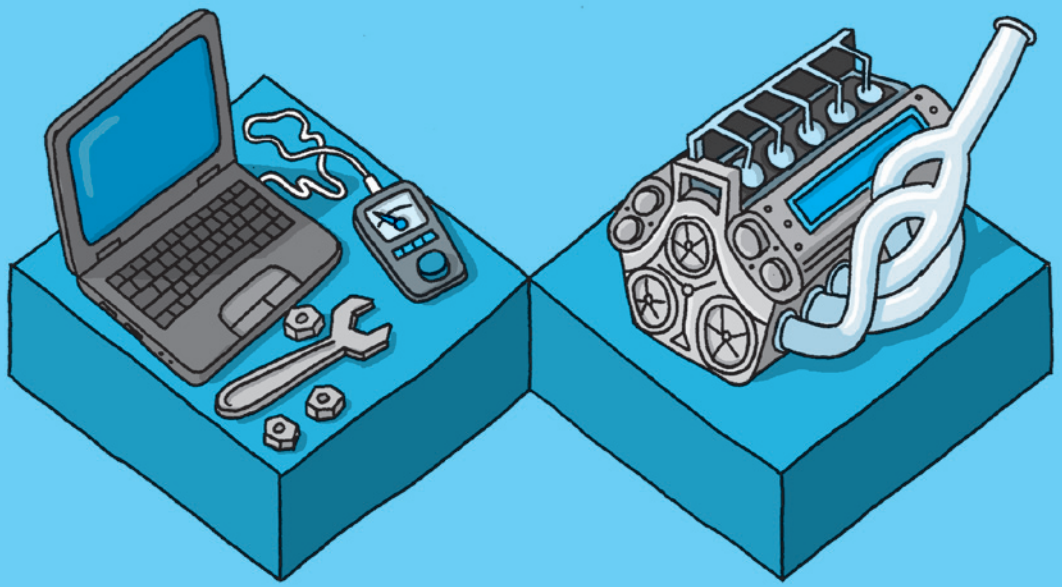
Consider negative or hindering impacts as well as positive ones.

Important actors can often help or hinder the outcome in many different ways. Once you discover the first impact of an actor, think about what else they could do.

EXAMPLES

- *Inviting more friends*
- *Purchasing tickets without calling the call centre*
- *Selling tickets faster*

WHAT?



DELIVERABLES

Once we have the first three questions answered, we can talk about scope. The third level of an impact map answers the following question: *What can we do, as an organisation or a delivery team, to support the required impacts?* These are the **deliverables**, software features and organisational activities.

Delivery plans and requirements documents are often shopping lists of features, without any context that explains why such things are important. Without a clear mapping of deliverables to business objectives, and a justification of that mapping through impacts that need to be supported, it is incredibly difficult to argue about making or not making an investment in certain items. In larger organisations with many stakeholders or product sponsors, this leads to huge scope creep as everyone's pet features and ideas are bundled in. No wonder such plans often fail.

An impact map puts all the deliverables in the context of the impacts that they are supposed to support. This helps with breaking deliverables down into independent chunks that provide clear business value, and helps us launch something valuable sooner. A clear hierarchy allows us to group related deliverables, compare them and avoid over-investing in less important actors or impacts. It also helps us to throw out deliverables that do not really contribute to any important impact for a particular goal. Finally, by connecting deliverables to impacts and goals, a map shows the chain of reasoning that led to a feature suggestion, visualising the assumptions of stakeholders. This allows us to scrutinise those decisions better and re-evaluate them as new information becomes available through delivery.

GETTING IT RIGHT

This is the least important level of an impact map. Don't try to make it complete from the start. Refine it iteratively as you deliver.

Treat deliverables as options, don't take it for granted that everything listed here will actually be delivered.

Don't go into a lot of detail early on, there will be time for that later. List only high-level deliverables. You can break down high-level features into lower-level scope items, such as user stories, spine stories, basic or extension use cases later. These items can become fourth, fifth or sixth level map branches.

Even on software projects, there are often ways of supporting a business activity without building software – sometimes it is cheaper to pay for advertising to recruit new players than spend months rebuilding a system. Consider anything that helps to achieve an impact.

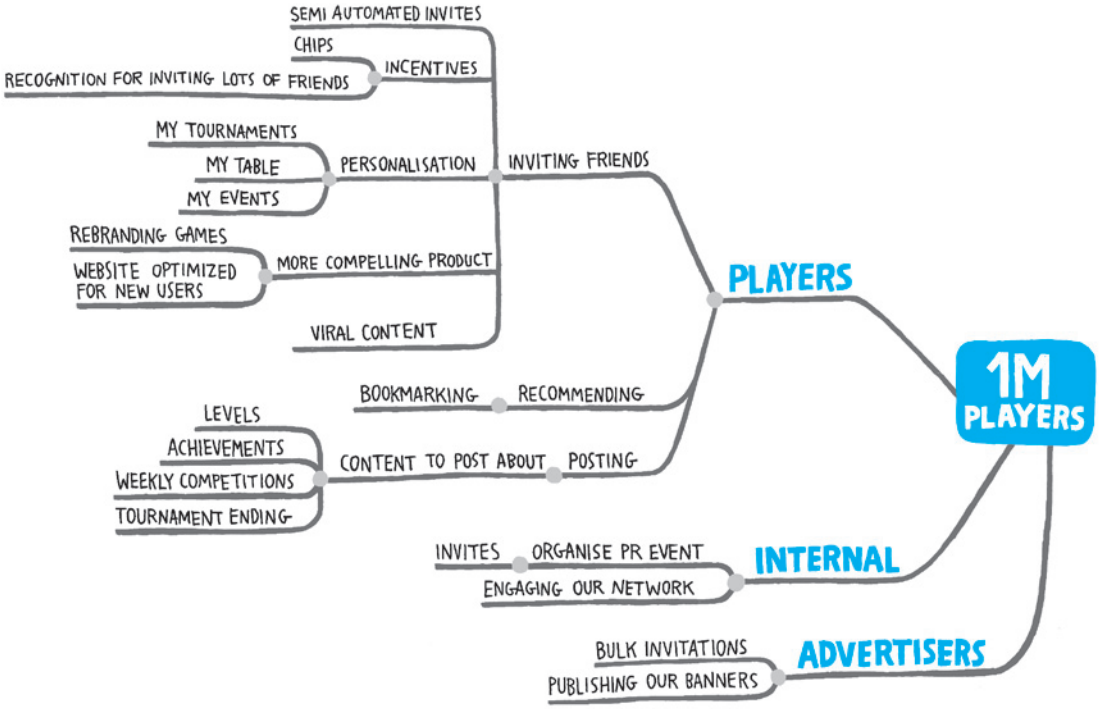
EXAMPLES

- *On-line ticket sales*
- *Mobile home page with purchase form*
- *Optimise call centre sales scripts*
- *Sign re-selling contracts*

Never aim to implement the whole map.

Instead, find the shortest path through the map to the goal!

EXAMPLE: ONLINE GAMING



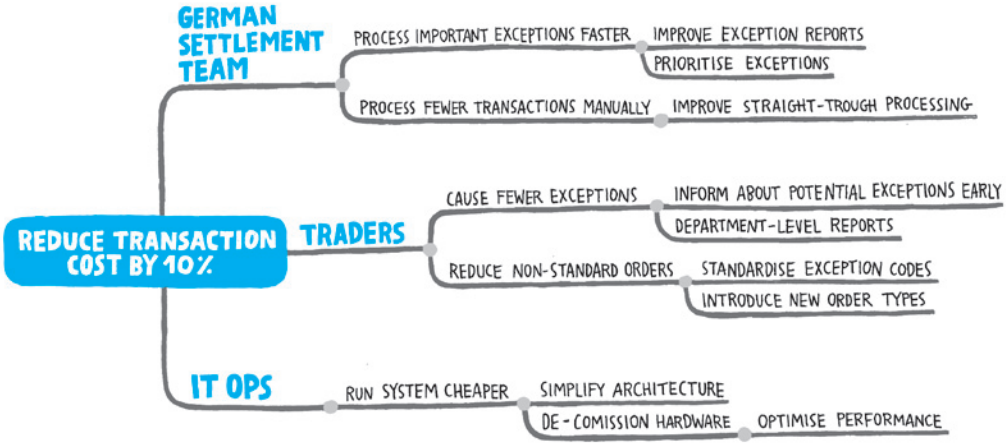
The map illustrates a milestone of an online gaming platform development. The key business goal for this milestone is to increase the number of active players to 1 million.

Players are important actors. They can help by recommending games to their friends, posting about the games on Facebook or inviting friends directly. Grouping those impacts together shows that they all contribute to the same goal, so we do not necessarily have to support all of them.

Another group of important actors are advertisers. They can also bring players by publishing adverts.

There are several deliverables that can impact player behaviour. For example, semi-automated invites can help make invitations more straightforward, a more personalised experience will make it more likely that players will invite friends, and reaching new levels or achievements should motivate players to post about the games on social networks.

EXAMPLE: FINANCIAL TRANSACTION PROCESSING



The map illustrates a milestone of a transaction processing system. The key goal is to cut the costs of processing by 10%.

The key assumption is that cheaper IT operations would bring this saving. Some deliverables that would help with this are a simplified system architecture and the decommissioning of older systems that require expensive support. These deliverables, though, would require a lot of work and any core architectural changes are high risk.

Another option is to reduce manual work required to process transactions. The settlement team in Germany is an important actor as they could reduce the cost of transactions significantly by processing the most important exceptions faster, which will reduce the work of downstream teams. Two potential deliverables could help: better exception reports and fully automated prioritisation of work. Another way to reduce cost would be to improve straight-through (fully automated) processing, directly reducing the workload of the settlement team and all downstream teams.

Traders creating orders are also important actors: they could reduce the cost of transactions by creating orders that require less manual work. There is an assumption that a significant amount of delay and unnecessary manual work is caused by the settlement team chasing after non-standard orders and free-text exception messages that are often just trader notes and not real exceptions. Perhaps we could reduce that. Standardising a list of exception codes and making traders input the codes directly instead of free-text notes is a business task, not a software delivery task, but could potentially be a quick and easy way to reduce cost.