Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Agile Business Intelligence
How to make it happen?

People matter, results count.

Table of Contents

1 Introduction

Over the past fifteen years, a number of adaptive techniques and processes, known as “Agile” methods, have gained much importance in IT and software development. Agile methods address the challenge of delivering high quality software products in market and business conditions that are perceived as unstable given the high pace of change.

Scrum is a framework for your organization to achieve Business Agility. This is your ability and flexibility to respond to - and even control- rapid evolutions inside of your organization and on your external markets. Scrum is the Agile framework that is most frequently adopted by organizations worldwide.

One of the areas in which Scrum can be applied is Business Intelligence (BI). BI environments are complex due to data focus with fast changing information needs and priorities, existence of many stakeholders, undefined timeliness, availability and quality of data, different systems to extract source data from, and continuously changing and emerging technologies.

By reading this document IT managers, Scrum masters, and BI specialists concerned with or starting to adopt Scrum, can learn how to deal with BI specific themes.

After a general introduction on Business Intelligence and Scrum, the specific characteristics of applying Scrum in a BI environment are explained. First is explained which expertise is required in a BI Scrum team: the Product Owner, Scrum Master and development team. Then is explained how user stories can be defined and the work can be fitted into sprints to realize BI product increments. Finally points of attention are listed regarding BI architecture, working in a distributed environment and working with partners.

Enjoy reading

2 Business Intelligence

Business Intelligence is the process of distilling information and knowledge from data. With this, companies try to find business insights that can help in running or improving the company. These business insights can come from different types of data and have different goals. Data sources are for example point of sale systems, purchasing systems, logistics, accounting, CRM and more recently big data and social media. The overriding theme is to support faster and informed business decisions.

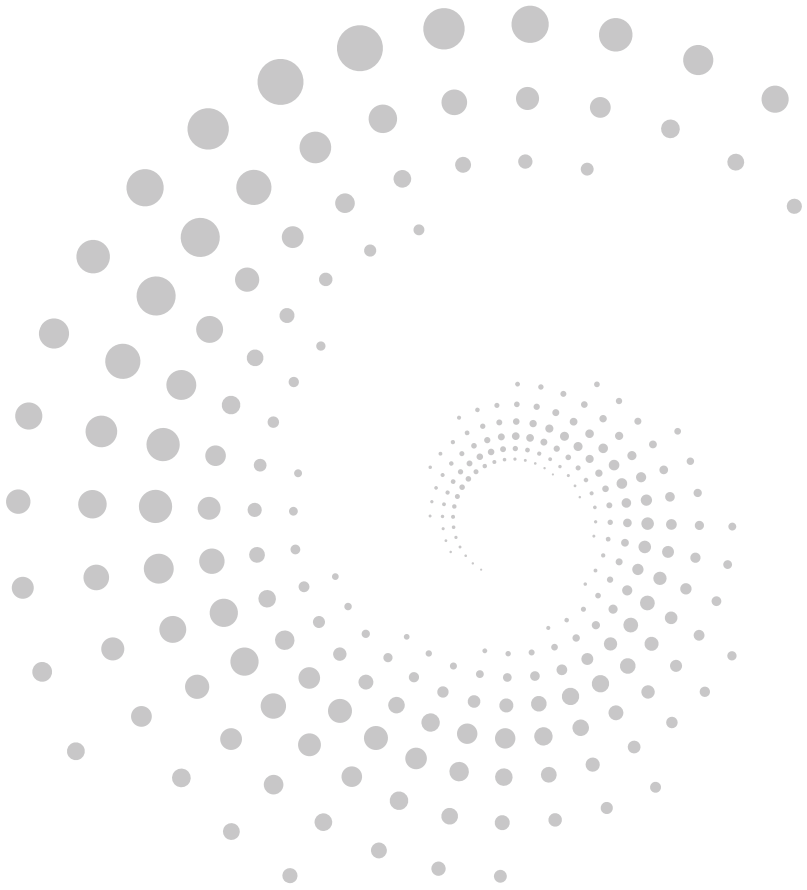
The goals of BI can be diverse, for example customer intelligence, financial reporting and consolidation, performance management, business process improvement, budgeting and planning.

The business insights can be disclosed via different data presentations like reports, dashboards and visualisations, or via analysis tools like OLAP or advanced statistical learning tools. Usually there are three different ways to incorporate the business insights in the business:

- Managed reports that are periodically refreshed.
- Self-service analytics.
- Input for operational systems (e.g. dynamic pricing or recomendations for a webshop)

Business intelligence also includes all activities to gather, prepare or decide on the necessary data and data definitions. Examples of these activities are:

- Data warehouse modelling.
- Data integration and ETL (extraction, transformation, load) for extracting data from sources and delivering it to target systems.
- Data governance for data definitions and data ownership.
- Validating data quality.



3 Scrum

More and more organizations discover that the only way to survive the current economic climate is to become more Agile. By being Agile, organizations satisfy their customers and employees. Customers are satisfied because products are delivered earlier, more frequent and contain the functionality customers ask for. Employee satisfaction grows by giving people freedom by letting them work in self-organizing teams.

This whitepaper is built around the Agile BI organization, using Scrum as a framework to deliver its value as the majority of organizations have chosen Scrum as leading framework. Note that there are several other Agile methods and frameworks like Feature Driven Development (FDD), Extreme Programming (XP), Smart and OpenUp. It is advised to include practices from these methods to your own whenever suitable.

An Agile BI manifesto has been created to uncover better ways of developing Business Intelligence by doing it and helping others do it. Through this work we have come to value:

- **Data and Information** over processes and tools.
- **Business Insights** over comprehensive documentation.
- **One team Collaboration** over contract negotiation.
- **Responding to change** over following a plan.

That is, while there is value in the items on the right, we value the items on the left more.

The “Agile manifesto” comes with very valuable principles supporting the improved way of delivering software products. These principles are also applicable when developing Business Intelligence products amongst which:

- The highest priority is to satisfy the customer through early and continuous delivery of valuable intelligence.
- A BI product must be delivered frequently, from a couple of months to a couple of weeks, with preference to the shorter timescale.
- Business people and developers must work together.
- Make sure to build BI products around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- Simplicity - the art of maximizing the amount of work not done - is essential.
- The best architectures, requirements and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.

Scrum is a framework in which you can address complex adaptive problems, while productively and creatively delivering products of the highest possible value. Many organizations use Scrum as a means to increase their Agility. Scrum can be applied to achieve an Agile BI organization too.

The Scrum framework consists of Scrum Teams and their associated roles, events, artifacts, and rules. Scrum employs an iterative, incremental approach to optimize predictability and control risk. Scrum is built around transparency, inspection, and adaptation. (Schwaber, Sutherland, 2011)

The Scrum Team consists of a Product Owner, the Development Team, and a Scrum Master. Scrum Teams are self-organizing and cross-functional. The **Product Owner** is responsible for maximizing the value of the product and the work of the **Development Team**. The Product Owner is the sole person responsible for managing the Product Backlog. The Development Team consists of professionals who do the work of delivering a potentially releasable Increment of “Done” product at the end of each Sprint. Only members of the Development Team create the Increment. The **Scrum Master** is responsible for ensuring Scrum is understood and enacted. Scrum Masters do this by ensuring that the Scrum Team adheres to Scrum theory, practices, and rules. The Scrum Master is a servant-leader for the Scrum Team.

Figure 1

Roles	Artifacts	Events
• Product owner • Development team • Scrum master	• Increment • Product backlog • Sprint backlog	• Sprint • Sprint planning • Daily scrum • Sprint review • Retrospective

The heart of Scrum is a **Sprint**, a time box of one month or less during which a "Done", useable, and potentially releasable product Increment is created. The work to be performed in the Sprint is planned at the **Sprint Planning Meeting**. The **Daily Scrum** is a 15 minute time boxed event for the Development Team to synchronize activities and create a plan for the next 24 hours. A **Sprint Review** is held at the end of the Sprint to inspect the Increment and adapt the Product Backlog if needed. The **Sprint Retrospective** is an opportunity for the Scrum Team to inspect itself and create a plan for improvements to be enacted during the next Sprint.

The **Product Backlog** is an ordered list of everything that might be needed in the product and is the single source of requirements for any changes to be made to the product. The Product Owner is responsible for the Product Backlog, including its content, availability, and ordering. The **Sprint Backlog** is the set of Product Backlog items selected for the Sprint plus a plan for delivering the product Increment and realizing the Sprint Goal. The **Increment** is the sum of all the Product Backlog items completed during a Sprint and all previous Sprints.

Figure 2: Scrum Artifacts & events

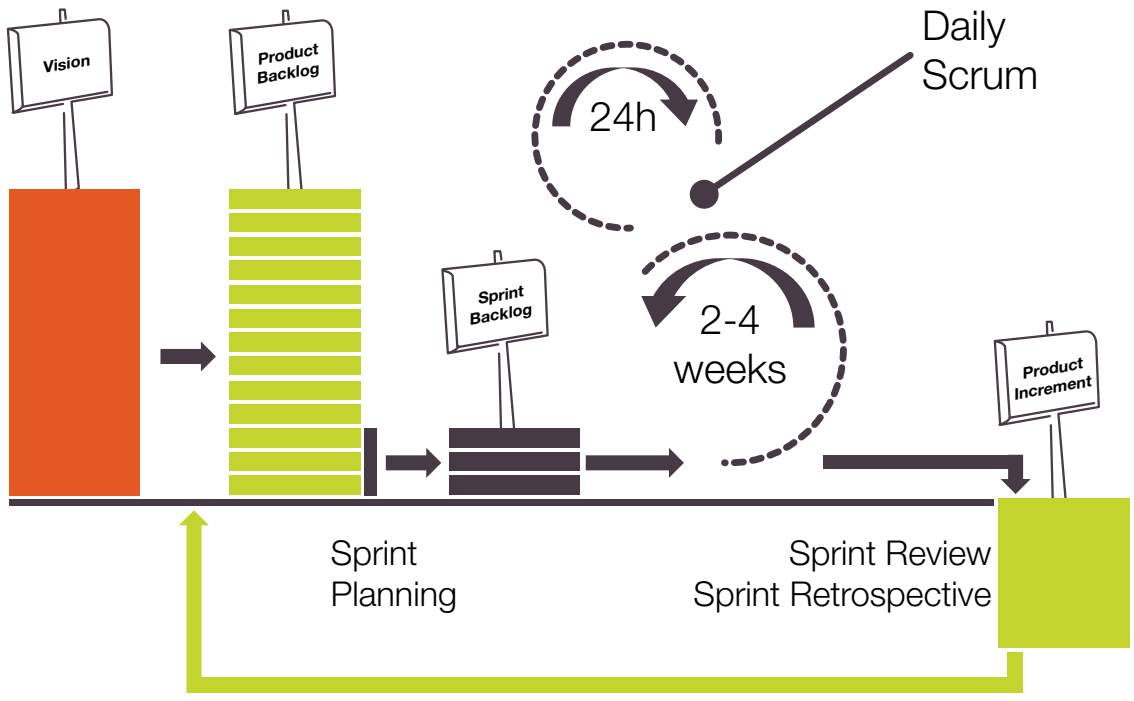
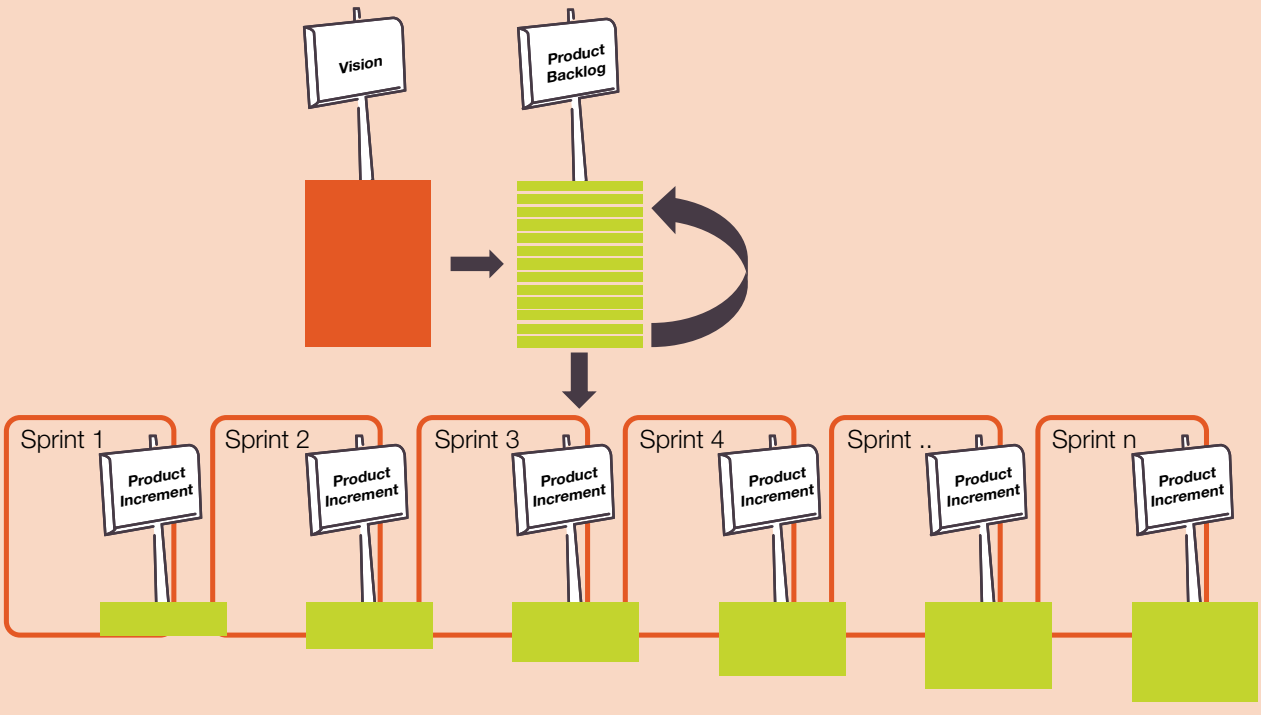


Figure 3: Scrum, building the product



4 BI Scrum teams

Teams are a highly underestimated productivity increasing factor. In Scrum, to achieve high performing teams, there are no fixed roles in the development team. Everybody in the team commits on achieving the results in a fixed period of two to four weeks together. To make a great team handovers of work must be limited because this introduces “waiting” time (waste). The people in the teams will grow accustomed to growing multiple skills and achieving a very high degree of collaboration with just enough paper.

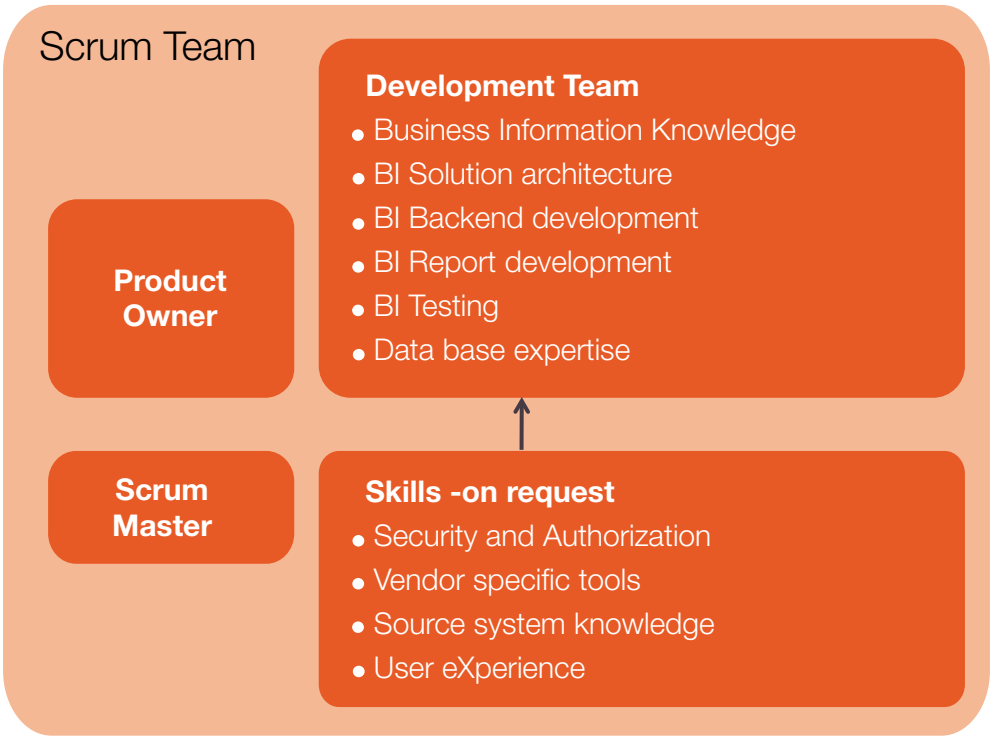
To create a high performing BI Scrum team, members of the Development Team require certain skills. BI skills differ from software development team skills. An overview of the skills required in a BI development team be found in the figure below.

Next to the Development Team also the Product Owner and Scrum Master skills require some attention.

The Product Owner

The Product Owner of a BI product development has to understand the typical BI development characteristics and the importance of the development for the business. The product owner must be aware of the different business stakeholders, who, for BI developments, may originate from different business departments, each with own goals. It is important that the product owner understands the priorities of the business users, orders them and manages their expectations. Feedback from the business users must be organized. This can be related to for example data quality, look and feel, or the analytic possibilities of the solution. Clear Sprint objectives and approach bring transparency to business users and development team about the expected feedback.

Figure 4: A BI Scrum team



The Scrum Master

The Scrum Master is responsible for good application of Scrum and continuous improvement of usage of the Scrum framework. A “BI Scrum Master” has to be experienced in Scrum and next to that understands BI and its typical characteristics amongst which, the importance of data quality, BI engineering principles, BI testing and release management.

The Development team

Why do BI projects require the development team composition as shown in the picture? BI is all about data, data gathering, transformation and reporting. Within the BI development process specific skills are key for success:

- BI Solution Architecture;
- BI Development: Back-end development, Database setup & Reporting.
- BI Analysis;
- BI Testing;

Knowledge of the business domain and the information relevant in this domain must be available in the team to ease communication with business stakeholders, challenge them and build the best possible solution.

The BI solution architect oversees the identified solution, is able to make choices and supports the team from a “technical” point of view (comparable to software architect in custom software development).

The BI developer requires the same skill set as any other software developer plus the ability to handle data. For a developer the focus on data also requires a different way of working. The developer needs to be able to integrate data on a large scale and perform transformations at the right moment.

The focus on data also means that the BI analyst is less process driven. The BI analyst focuses on the usage and transformation of the data. This entails the collection of data from different source systems, integrating the data into a data warehouse and presentation in reports.

The BI tester has to be experienced in Agile BI testing practices and how these can be applied in Scrum teams. The BI tester needs to understand the use of the solution and the importance of the data flow through the solution. Each data transformation must be tested individually in order ensure that the end result and output of the transformation is correct. The challenge in Agile BI developments is the creation of representative test sets and test automation.

Members of the development team are multi-disciplinary and are able to support each other to reach the Sprint objectives of the development team. Does this mean that everybody is an expert on everything? No, but the ability to oversee each other’s role and step in to support one another improves the team productivity. The idea is to create the so called “generalizing specialist”.

Scrum teams may need skills for specific topics. Often required are:

- Security and authorization specialists to be able to define different user roles with variations to data access.
- Source system specialists to explain data definitions and how transaction and master data can be extracted and loaded into the BI solution.
- Vendor specialists to support the development team on tool specific questions.
- UX specialists, who can advise on the usability of the solution.

While detailing user stories these skills can be identified and made available to the team on the right moments.

5 User Stories in BI systems

One of the most asked questions in Agile BI is "how can we split the work for a report or a data warehouse into small chunks". The answer to this question is key in the understanding of how Agile can be successfully applied to BI. Therefore a BI user story model has been developed, that can help in answering just that question in real BI projects.

In his book "User Stories Applied" (Cohn, 2004), Mike Cohn defines a user story as "... describes functionality that will be valuable to either a user or a purchaser of a system or software". For a BI system the value itself is not in actions the software performs itself, but mostly in the information the system delivers and what the user can do with the information. So in the Business Intelligence world, user stories will most of the time be about information.

Of course for BI user stories, the same rules apply as for user stories in other software development projects. Think about INVEST (Independent, Negotiable, Valuable, Estimable, Sized appropriately, Testable), prioritized on the product backlog, delivered according to acceptance criteria, and so on. Remember that there is more value in BI than only in reports. In BI systems, five types of user stories are distinguished:

- Data disclosure stories.
- Data presentation stories.
- Data augmentation stories.
- Data validation stories.
- Configuration stories.

Data disclosure stories are about extracting data from the source system and making it available in self-service BI environment. While it feels counter-intuitive, in these stories it is very useful to mention the source system. Most of the times the user has a prevalence for a certain system, that he/she already uses in his/her daily work. Adding this information to the user story makes it clear what the user actually expects, and can at least help to get the conversation going about what is the best source for this data. However, don't make a user story too large. If the source system contains several information items, e.g. costs, revenues, purchase invoices and sales orders, then it's better to keep the stories confined to a single information item. A few examples:

- "As a sales analyst, I want to be able to analyze with revenues on sales transactions from the Point-Of-Sales system on a weekly basis, so that I can find patterns in sales over time."
- "As a financial controller, I want to be able to view all the outstanding credit lines from the Loans-system on a monthly basis, so that I can verify the balance of the department."

Data presentation stories describe the presentation of the information in a format that the user can easily understand. In order to use the information, the user needs to be able to draw conclusions, and explain the conclusions to others. These can be (cross)tables, graphs, infographics or outcomes of (statistical) analyses. Some of these user stories will be fairly simple. Creating a crosstable based on data readily available in a data mart is usually not much work. On the other side of the scale you have complex statistical analyses to test hypotheses about customer behaviour. These can be very complex and will take a lot of time to produce.

- "As a sales manager, I want to have a monthly report, which shows me the sales figures by sales agents, so that I can steer the agents to improve the sales."
- "As a marketing manager, I want to know whether customer age has a relationship with the product segments bought, so that I can find smarter ways to market our products."
- "As a corporate risk manager, I need to have a monthly report on all Risk Weighted Assets for my bank, so that I can report to the regulator on our positions."

Data augmentation stories are about creating new information based on already existing information. For these derivations business rules are used. You can derive profit margin based on costs and revenues. You can derive customer age segment based on his date of birth. And if you have done sufficient analysis, you can derive the chance of customer churn with a calculation model that uses for example recent transaction behaviour, website usage and number of helpdesk calls.

- "As a call center sales agent, I want to see the customer age segment, so that I can use the communication style and offers that are most likely to fit the customers taste."
- "As a customer account manager, I want to see the customer churn probability for a customer, so that I can focus on the clients that are most likely to leave."

With **Data validation stories** a user talks about applying business rules to check whether the data that is extracted, is of good enough quality to be used for the end users to work with, and to take the necessary actions when the data is not good enough. Not all actions will be committed automatically, but if not, at least a certain user needs to be notified that anomalies exist.

- "As a compliance manager, I want to see whether the complaints report is based on all complaints that are issued during the last month, so that I can send the report to the regulator without breaching the rules of the regulator."
- "As a data steward for the loans department, I want to be notified when mortgages exist, that do not have proper customer data attached, so that I can take proper actions to keep risks as low as possible."

Finally, **Configuration stories** are about enabling maintenance staff and administrators to keep the configurations of the BI system up to date, without having to change the code. This can be for user authorisation and security settings, reference data or hierarchies for which no separate source system exists, and so on. Quite often the user in the user story is not a business user, but a user from the IT department, doing maintenance on the system.

- "As a product manager, I want to be able to configure and update the product hierarchy, so that I can keep the hierarchy up to date for correct analyses."
- "As a security administrator, I need to be able to configure which users are allowed to see information for which branches and regions, so that I can assure compliance to the data confidentiality rules of the company."

Most of the time these types of user stories will be combined. For example, a monthly report on outstanding risks usually consists of numerous tables and graphs. Besides, the report needs a lot of source and derived information to be available in a data mart, and the quality of this information also needs to be validated. Before you have finished this report, you will have been working on at least four different types of user stories, and probably more than one of each type. While it is easy to think that in this case the report, with all underlying data streams and business rules, is the smallest denominator of work, each of these stories present an amount of value to the end user, and therefore can be delivered separately.

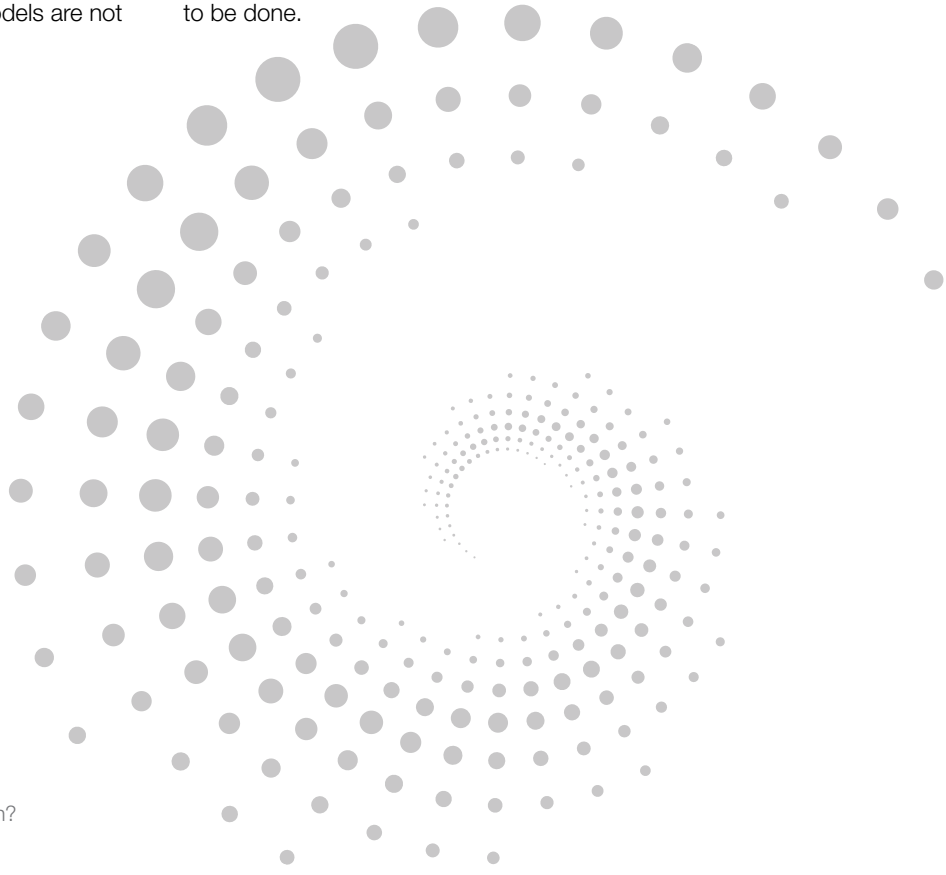
6 How to fit the work in sprints

Now it comes to fit the work for the stories into sprints of two to four weeks. For BI that is a challenge. If we see the report as the smallest piece of work, we want to plan all the work for this report in one sprint. And next sprint we will take up another report, and so on. However, you will find that will not work optimally. Quite often, all of the work for a single report can be that much, that it will not fit in a sprint. Even more important: if at the end it turns out that the report isn't what the users expected, you will face a lot of rework in the whole chain of dataflows you created to make the data available, through the staging, data warehouse and data mart layers. You would liked to have that feedback earlier, before doing a lot of work on the back-end.

What is the alternative? Every now and then you see a BI team go back to the old idea of splitting the work according to the layers of the data warehouse architecture. First extract the data to the staging, next sprint load it to the DWH-layer, after that load it to the data mart and in the fourth sprint create the reports. Obviously it will become much easier to fit the work in the sprints. But it didn't solve the problem of the late feedback. Contrary, instead of after one sprint, you now only get the feedback after four sprints. An important attention point in Agile BI is fast and early feedback. So clearly these two models are not going to help us in becoming Agile.

Luckily there are other options. The first option is to do some prototyping for the report (see figure 5). Not just movie studio type cardboard prototypes, but working prototypes that show real features and real data, like they will be in the to-be report. For this it's best to extract the data into your staging tables (but not further!), and create "dirty" queries that fill the report. That is enough for the first sprint. At the end of the sprint you can demo the report to the users, and get feedback. While prototyping, think about what kind of feedback you want to get. If you want to get feedback on the report lay out, use mock-up data that is clearly wrong. When you want feedback on the correctness of the data, leave out the lay out, use real production data, and focus on the numbers.

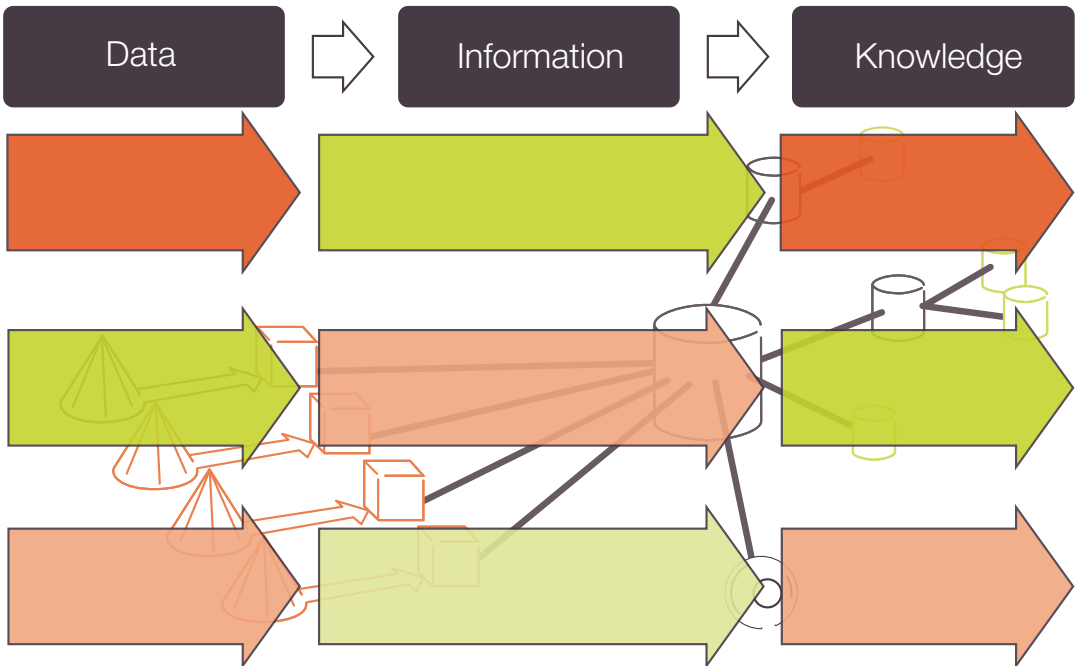
When the prototype looks acceptable for the business, you can move on in the next sprint to create the dataflows and data-models in the layers between staging and report. Of course this method requires strong management of architectural and maintenance guidelines. You don't want to risk that the prototypes become the final solution. They will be very hard to maintain and performance will probably be low. So keep making that clear for the business sponsors, so they understand that after the prototype report is accepted, still some work has to be done.



For a second method, we get back to the different types of user stories. For every report, decompose the work into stories of the different types. If possible multiple stories per type (e.g. costs and revenues as separate data disclosure stories for a sales/profit report). With this list of stories, start with the ones that you expect most feedback on and the ones that are the most risky. These are the Data presentation stories and the Data disclosure stories. Within these decompositions if further breakdown is required, consider the need to deliver key dimensionality throughout prior to key user measures. For example, if a use requires profit by region and by client and within itself this would be exceptional to deliver in a single sprint, consider delivering the region or client hierarchy first and completing with the profit measure. Leave the stories that have lower value and lower risk for the next sprint:

- Data augmentation (e.g. calculate the profit margin based on costs and revenues).
- Data validations.
- Configuration.

Figure 5: Prototype data warehousing



Source: Hoogendoorn, 2012

7 Agile BI Architecture

When working on a BI product, it is important to also apply the Agile principles and values to the architecture:

- Welcome change: make sure that you architecture supports new or changing requirements.
- Working software: when you have the idea that something "should work", make sure that you get to know soon that it really works as expected.
- Simplicity - the art of maximizing the amount of work not done - is essential.
- The best architectures, requirements and designs emerge from self-organizing teams.
- Continuous improvement: always seek ways to improve the architecture.

Now this is very similar to architecture in any other kind of software development project. Handling the software architecture and performing architecture related activities is the same. What stands out in BI and DWH projects, is how to start. When you start to work Agile in an existing project, or with an existing architecture, you will continue with the current architecture, but handle it in an Agile way. However, when you start with a new system, and your architecture is a clean sheet, you also need to start Agile. It is tempting to try designing the complete architecture upfront in detail. But does that give you a good start in welcoming change later on in the project? High chance that later on in the project you will find that one of the decisions you made early on now blocks the changes you want to make. And large chance that you could have kept options open until now. On the other hand, it is also easy to think that all architecture will come in due time, and that no decisions have to be made upfront. However, it is difficult to build a decent house on quicksand.

So you have to set a starting point for your architecture. In BI projects, there are a few decisions that are best made early on in the development, because changing them later on will have huge costs:

- Choices for tooling, e.g. database, ETL-tool, reporting tool, data modelling tool, version management tool etc.
- Choices for datamodel – method (3rd Normal, Start scheme, Data Vault etc.).
- Choices for standard ways of processing, e.g. historic data processing and error handling.

If really needed, you still can change these decisions later on, but the costs will be large. You probably have to rebuild large parts of your architecture, so be careful, and make smart decisions.

An important part of the BI architecture of course is the datamodel. For this the same applies as the rest of the architecture: be careful with a big design upfront. You will find out later, that some entities should have been modelled differently, or that certain attributes are not needed. Also, make enough decisions to be able to start. Like described above, first choose a method of data modelling, then determine the most important entities. For these entities, determine the functional keys and the most important relations. Finally, take a bit of time to decide on an approach for the most important technical metadata fields (e.g. surrogate key, load time, valid from and valid to dates, etc.). That's all you need for a starting point. All other details (non-key attributes, entities and relations of lower importancy, other metadata fields etc.) can and should be added when they are really needed.

Be very careful with adding entities and attributes for "just in case" or "then we already have that when we need this in the future". Most of the time you will not need them in the future, or when you do need them, you need them differently. Agile architects usually use the acronym YAGNI, short for 'You Ain't Gonna Need It". Experience tells that almost always, this is really the case. In Data Warehouses, it is quite often even worse: there is a table that was added long time ago, "just in case", and needs to be used now. After analysis, it turns out that the data in there is useless, because the source extraction had been wrong all the time. Now the effort to get the correct data is even larger than it would have been when the table wasn't around. So avoid adding elements that are not needed or used immediately. In the same way, remove elements that turn obsolete over time.

8 Distributed Scrum, Scrum and Offshore done right!

In many organizations, BI development teams are spread over multiple locations. Having teams distributed over more locations gives specific challenges:

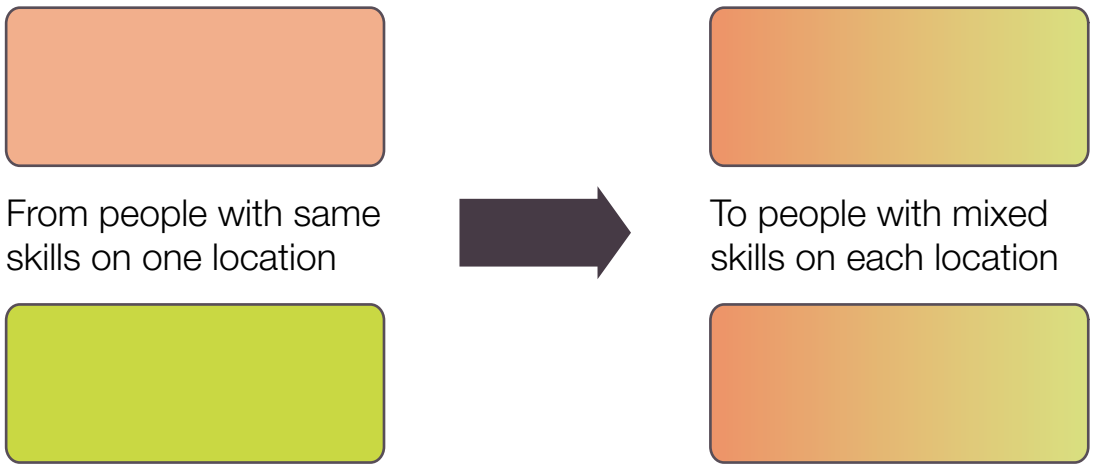
- Ineffective collaboration and communication: there's no smooth collaboration and efficient communication between teams on different locations, resulting in waiting times and waste.
- Demotivated teams: teams lose their motivation by waiting for other teams or team members that are working on "higher" priorities.
- Unclear team responsibility: no agreements are made for example regarding unit and system testing, data conversion, configuration management or tool support.
- Lack of knowledge: knowledge of data and how data is used by its users is key in BI developments. The required knowledge of the people in the offshore teams is often too low resulting in mistakes and miscommunication.
- Handover warfare: dropping intermediate results is an inefficient process leading to miscommunication and results in additional effort.
- Unclear status and progress: it is not clear what the status and progress of the teams is. Products are always "almost done" and teams can't predict when it is really finished.

Experience with distributed teams working in an Agile manner with Scrum as framework shows that "Distributed Scrum" is successful when it is applied in its entirety, without concessions. The distribute team works together with the business representative as one team, transparent for the environment, and delivers working products continuously. The critical success factors:

- Continuous improvement, inspect and adapt: for distributed teams, it is essential to deliver in every Sprint and to improve in every Sprint. Early and continuous feedback on the delivered product leads to working software that is in line with business expectations. Inspection and adaption of teams' own performance leads to improved collaboration and communication.
- One team: work as one team on the Sprint goals, as agreed upon during the Sprint Planning. Continuously help each other, even if people are located over different locations. Together reduce the amount of work that is not done. A good experience is to distribute people with different skills over the locations (see figure 6).
- State-of-the-art collaboration tools: involve each team member in each Scrum event and facilitate peer-to-peer communication by using state-of-the-art collaboration tools, video-conferencing, teleconferencing, chat functionality and information sharing functionality. For example organize daily standup meetings with the entire team. It is essential that the scrum board is visible to all and that there is a common repository.
- Knowing and seeing each other: our best practice is that face-to-face contact and knowing each other is essential to achieve optimal team results. To accelerate this, the team must be brought to one location during the first Sprints of the project. The team then has also the possibility to address cultural differences. After the "on-site" period team communication via collaboration tools goes much more smoothly.
- No team leads: there are no project leaders or team leaders within the team. Members of the development team communicate and collaborate with each other wherever they are. Only then creativity, commitment and responsibility is at the level where it should be, the development team.
- One team: a Scrum team is built out of a group of multi disciplinary people, working together on a basis of equality. There is no front-office versus back-office team nor is there a main team versus a test team. People of a Scrum team help each other during each step of the delivery process.

- Third party involvement: people of third parties are integral members of the Scrum team. Contracts with these parties may not influence Scrum practices.
- Team member contribution: the entire development team should be involved in every Scrum event. Ensure an appropriate meeting location that is equipped with proper conference tooling so that each member, wherever located, really participates.
- Respect the Scrum framework in its entirety: all events, roles, artifacts and rules. This way, a culture of continuous improvement starts to grow, allowing the team and stakeholders to realize all benefits a distributed Scrum approach can bring.
- Experienced Scrum Master: taking all previous bullets into account, in our opinion, an experienced Scrum Master is key to realize successful distributed Scrum teams.

Figure 6: Distributed Scrum teams



9 Working together with partners

“One team Collaboration over contract negotiation”. Many BI development organizations use partners to help, guide and support them in achieving the best BI solutions. When moving away from the traditional way of BI development to Agile BI it’s important that these partners adopt themselves in the same way.

Criteria for successful relationships:

- The partner must be trained and experienced in Scrum.
- The partner must have the possibility to deliver Scrum Masters and all competencies as mentioned in the Scrum Team setup.
- Consultants must be trained to work together with other members of the Scrum team. Different origins of the team members may not be noticed. “Sub teams” should not be created.
- Knowledge of the business domain and experience in the IT solutions used.
- Long term allocations of consultants to the customer.
- When distributed teams are required, state-of-the-art communication facilities must be available. See also the section about Distributed teams.
- Manage the contract outside the visibility of the people in the Scrum teams.

These criteria for successful relationships are not specific for BI teams.

10 Conclusion

Agile methods address the challenge of delivering high-quality software products in market and business conditions that are perceived as unstable given the high pace of change. Regarding BI products the same conditions apply as BI environments are usually characterized as complex due to fast changing information needs and priorities, existence of many stakeholders, undefined timeliness, availability and quality of data, different systems to extract source data from, and continuously changing and emerging technologies.

Adopting Agile practices in a BI environment is needed to deliver successful BI products. It was experienced that the Scrum framework can be applied in its entirety in the field of BI and data warehousing. Combined with the Agile mindset and disciplined BI engineering practices, this creates great value for organizations and its customers.

Specific points of attention when applying Scrum in a BI context are the composition of BI Scrum teams, the BI specific user story types, how to fit them into Sprints and the architectural considerations.

A challenge for the future will be to incorporate development practices like automated testing, automated deployment and continuous integration to the BI landscape.

When applying Agile in a BI environment, the keywords turn out to be: business value, early feedback, collaboration and continuous improvement.

References:

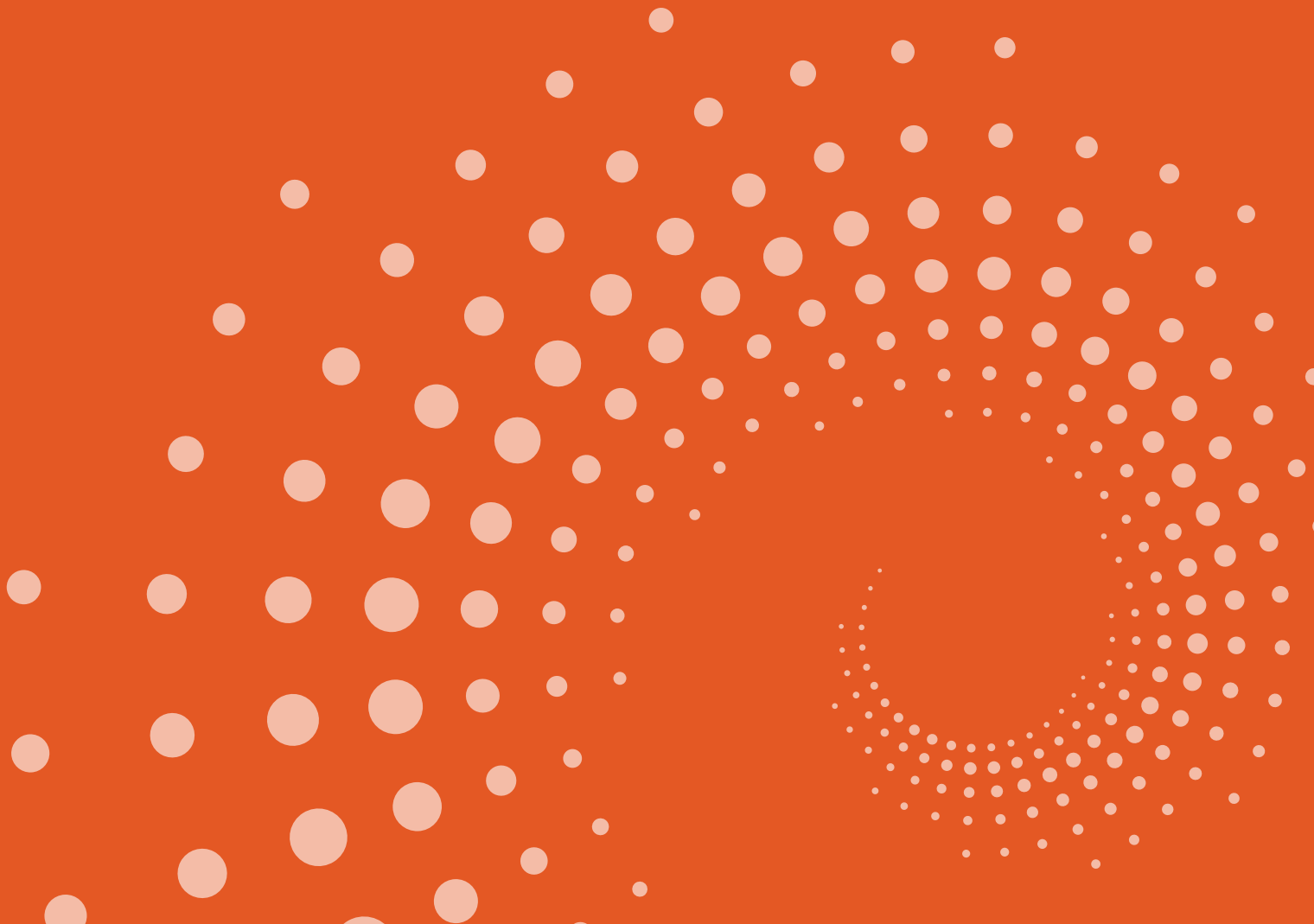
- Cohn, 2004: Mike Cohn, *User Stories Applied for Agile Software Development,* 2004. Addison-Wesley.
- Hoogendoorn, 2012: Sander Hoogendoorn & Sandra Wennemers, *De Voordelen van Scrum en Smart,* Maandblad Informatie, December 2012.
- Schwaber, Sutherland, 2011: Ken Schwaber & Jeff Sutherland, *The Scrum Guide,* October 2011, www.scrum.org.
- Manifesto for Agile Software Development http://agilemanifesto.org

For more information contact:

Jorgen Heizenberg
Principal Technology Officer
jorgen.heizenberg@capgemini.com
Twitter@jheizenb
nl.linkedin.com/pub/jorgen-heizenberg/1/831/234

Arjan van den Berk
Managing Consultant
arjan.vanden.berk@capgemini.com
nl.linkedin.com/in/arjanvandenberk/

Renze Feitsma
Senior Consultant
renze.feitsma@capgemini.com
nl.linkedin.com/in/renzefeitsma/



Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

About Capgemini

With more than 125,000 people in 44 countries, Capgemini is one of the world's foremost providers of consulting, technology and outsourcing services.

The Group reported 2012 global revenues of EUR 10.3 billion. Together with its clients, Capgemini creates and delivers business and technology solutions that fit their needs and drive the results they want. A deeply multicultural organization, Capgemini has developed its own way of working, the Collaborative Business Experience™, and draws on Rightshore®, its worldwide delivery model.

Learn more about us at

www.nl.capgemini.com/BIM

People matter, results count.