IBIMA Publishing
IBIMA Business Review
http://www.ibimapublishing.com/journals/IBIMABR/ibimabr.html
Vol. 2011 (2011), Article ID 685309, 9 pages
DOI: 10.5171/2011.685309

The BONSAI Approach Applied to Small-Medium Enterprises

Diego Bruno

Torino, Italy

Abstract

In this paper, we describe the BONSAI approach and how Small-Medium Enterprises can take advantage of it. BONSAI consists of a set of behaviors, best-practices and guidelines which help professionals to develop better software. Conceived as an iterative and incremental process, it helps to integrate the software development process with the upper processes of the enterprise taking advantage of Business Process Management techniques. It brings to Small-Medium Enterprises the capability to support continuous changes helping them to mitigate and assess the risks involved.

Keywords: Agile, Business Process Management, Small and Medium Enterprises

Introduction

Today, a large number of companies from different commodity sectors are developing software. From the "one-man-band" company to the larger international manufacturer, they develop very different software and very differently in terms of markets, products, technologies, and of course outcomes.

Small and Medium Enterprises (SMEs) are a group of those. Following a commonly accepted definition, an SME is a company with a number of employees ranging from 1 to 500 people. Compared to Large Enterprises (LEs) where people are several thousands, it seems to be just a bonsai compared to an oak.

But the definition of the above mentioned SME is not developing very accurate for companies in which software is part of the main business. To be more specific, let's consider software projects as a meter: we consider an SME when the average of in-house software projects developed, or in progress to be developed, during one year ranges from 1 to 10 software projects.

Also, SMEs have several key differentiators from LEs. Following the Ibrahim and Goodwin study (1986), "Small businesses are known to suffer from resource poverty. Resource poverty is characterized by immense constraints on financial resources, a lack of human resource expertise, and a short range management perspective imposed by a volatile competitive environment.". In this sense, SMEs have restricted investment capabilities caused by financial limitations: usually, SMEs cannot take advantage of economies of scale, incurring high costs.

Compared to LEs, we should notice that SMEs have more dynamism and flexibility: they are more responsive to changes and new ideas, especially because the hierarchical structure is shallow. But specialized staff are lacking in SMEs: they can rarely afford a specialist for each enterprise area. They don't usually implement company-level Development Methodologies and Formal Processes for the

development of software, mainly because of a lack of resources in terms of budget, people, knowledge, internal communication, and they can hardly afford such investment. As a consequence, every Business Unit usually adopted its own way to develop software.
It should also be noticed that in the last 10 years they have followed the global trend to offshore the production activities, and in the last 5 years they have started to offshore software development to both offshore companies and overseas subsidiaries.

Looking at these differences, it's interesting to realize that SMEs seem to have a profitable environment for improvement of software development, but often they don't have the necessary resources to perform the change. The main reason is that improvement is a cost, often not affordable for SMEs. Relevant costs include: reorganization costs, education, scouting and selection of development methodologies, development methodology validation (including pilot projects), process implementation, tool licenses and maintenance. This is a key differentiator from LEs, which can usually afford these costs without interfering with the main business of the company.

But in which sense is it useful to compare an SME with an LE? Which Software Development Processes can be adopted by SMEs or at least inspire them? Which ones should be modified and optimized in order to fit SMEs requirements? Finally, what can BONSAI do about developing better software projects?

Being based on the experience of SMEs in the past 10 years, the BONSAI approach tries to answer those questions.

The Role of Business Process Management

Business Process Management is a discipline widely adopted by LEs. To describe how to improve enterprise software processes, BONSAI makes large use of the Business Process Management approach (BPM). Following the Di Leva and Laguzzi (2008) definition, "Business Process Management is a structured approach, founded on a group of activities required to design, optimize, monitor and integrate enterprise processes, in order to create a process aiming to make the enterprise business efficient and effective."

In the sphere of Business Process Management, we decided to represent enterprise processes as aggregation of Business Building Blocks (BBBs). Following the BETADE project definition (Verbraeck and Dahanayake, 2002), "The building blocks are a self-contained (nearly-independent), interoperable (independent of underlying technology), reusable and replaceable unit, encapsulating its internal structure and providing useful services or functionality to its environment through precisely defined interfaces. A building block may be customized in order to match the specific requirements of the environment in which it is used (plugged)." In other words, BBBs are independent, interoperable (that is independent from underlying technology), reusable and interchangeable.

Additionally, BBBs have an internal structure, they supply services or functions through an interface, they are customizable and adaptable to every specific need, they embrace the concepts of inheritance and encapsulation of their internal structure.

It should also be noted that BPM and especially BBB concepts are very similar to Object Oriented concepts, well-known in Object Oriented Programming Languages like ADA, C++, etc.

Definition of BONSAI

So far, BONSAI is not a Development Methodology, like waterfall, V-model, RUP, etc. It aims to define a set of guidelines which helps professionals to develop better software. It is a collection of approaches, best-practices and guidelines. Whereas it has his roots in SMEs, it is not aimed to be necessarily applied by SMEs only.

BONSAI stands for: **B**usiness-**O**riented **N**ormalized **S**ynergic **A**daptive **I**teration.

The term Business-Oriented refers to a particular attention of the development process towards the whole enterprise business. In this sense, the software development process has to be constantly compared against the business needs and operations of the company, and software development processes are BBBs of a more complex Business Process, involving the whole Enterprise.

The term Normalized refers to both simplification and optimization of processes. Business Processes coming form LEs can hardly be successfully applied on SME, they need first to be normalized. Some normalization techniques will be described later.

The term Synergic encompasses several aspects of Software Development, including communication, people organization and personal motivation. A collection of guidelines aiming to improve synergy will be provided later.

The term Adaptive encompasses two aspects of Software Development. The first is about Enterprise Organization, and how organization influences Software Development. Later, it will be discussed how the Development Process can adhere to the implicit and explicit Business Processes of the organization. The second is about how to manage changes, that is how a process is able to adapt to variations from the initial inputs. Changes include rescheduling, specification changes, business plan changes. Some techniques to support changes inside processes will be discussed later.

The term Iteration underlines the iterative approach (versus waterfall approach) of BONSAI. Iteration is the base of many software development methodologies. We will refer especially to Iteration as defined by AGILE Modeling (Ambler, 2002). The BONSAI iteration, implemented as a BBB, will be detailed in the next section.

Iteration

Iteration is meant as a repetition of a particular task or activity with the goal of improving its main work. This method is often compared to the waterfall model, which is a sequential software development process, where progress flows downwards. Since SMEs have financial constraints, iteration is usually the preferred methodology. As a further consequence, for complex projects, the waterfall approach is most likely to fail for SMEs.

The core of the BONSAI approach is represented by a specific work unit, conceived as a BBB, called the Core Building Block (CBB). CBB represents the formal iteration of the software development process.

To understand how that iteration works, and how CBB behaves, we first describe the actors involved in the iteration. They are: the Developer Core Team, the Software Test Team (which includes both the Module Test Engineer and the System Test Engineer), the Tool Support Team, the Integration Team and the Project Management.

The Developer Core Team represents the group of people developing a specific feature. It is usually composed of at least Senior Engineer and several Junior Engineers. It is definitely the productive part of CBB, since they develop the more valuable working product of the CBB: the software needed to implement a given feature of the product. As Mills (1971) underlines, the Senior Engineer (or Chief Programmer) is seen as a surgeon.

The Software Test Team, Tool Support Team and Integration Team have the role of supporting the Developer Core Team to develop their work product: the Software Test Team develops and executes tests at software modules and/or at system level to assure that the quality of the work product is met; the Tool Support Team develops and maintains the necessary tools for development, such as the Configuration Management System, Version Control, Bug Tracking System, Knowledge System, tool-chains, Automatic Compilation System, etc; the Integration Team is responsible for putting every working product together, checking that the desired behavior of the system is met, and correcting, or asking for correction, of unwanted effects.

Project Management is finally the glue between each component. It supports each team, in terms of communication, scheduling, people management, synergy improvement, but it is also the interface between the Enterprise and the CBB.

Let's now have a look in broad terms at the BONSAI iteration. First, let's define the inputs and the outputs of the iteration. Inputs are represented by product requirements, coming from stakeholders. Outputs are represented by specific work products: prototype software, demo software, software release. The BONSAI iteration is composed of five sub-iterations.

The first sub- iteration is represented by requirement validation: stakeholders create/modify a Business Plan, then, elaborate requirements and priorities; then, the CBB validates requirements and eventually asks stakeholders for a refinement of requirements.

The second sub-iteration is about estimate validation and plan elaboration: the CBB creates cost and effort estimates; then stakeholders approve or disapprove, then they modify the list of requirements and priorities accordingly; finally the CBB creates a task list and a plan based on final requirements and priorities. If a task list is already available, it is refined based on previous interactions.

The third sub-iteration is about module development: for each task identified in the second sub-iteration, the software is developed; then software is validated by Module Test Engineers; finally defects are corrected by the Development Core Team.

The fourth sub-iteration is about integration: Integration Engineers integrate the modules; then the defects are fixed by the Development Core Team.

The fifth sub-iteration is about the system test: System Test Engineers validate the integrated software; then the defects are fixed by the Development Core Team.

The output of the iteration is then validated by stakeholders. Then, iteration starts again from sub-iteration 1.

Business-Oriented

For Business-Oriented approach, we mean that BONSAI is devoted to supporting the enterprise business. For that reason, the BONSAI CBB cannot be estranged from the upper enterprise processes.

In fact, each CBB iteration has direct and indirect impacts on several aspects of the enterprise business.

The first impact is about the level of confidence and knowledge of the project earned at each iteration. An estimate at the beginning of the project can be refined, and so the efforts and the delivery dates can change: this could impact especially on the original business plan.

Another impact concerns the change of team engineers, turnover, and overall high workload of engineers. It can change the scheduling and can impact stakeholders' expectations.

The Quality acceptance department is impacted since new software is ready to be tested at the end of each iteration. The Sales department and customer support are also impacted since a new release is ready to be sold, and a new release on the market must be supported. Production and the supply chain can be impacted since a new software is now ready to be finalized (e.g. burnt on a CD) and shipped to the customer.

Additionally, the enterprise itself can embrace changes that impact on software development: change or addition of new requirements from Marketing, changes coming from the field (e.g. low customer satisfaction, which can jeopardize the future business, can question the quality of the software produced, and so an optimization of the software development process can be needed), and changes in the structure of the company (e.g. department splitting or department merging).

To face all those impacts, an enterprise-global approach is needed. In this sense, CBB should be conceived as a single BBB of many other enterprise business processes. According to the BONSAI approach, all involved enterprise business processes should be orchestrated and integrated with CBB, at each iteration.

Since SMEs are usually more dynamic than LEs, and most LE Departments are simply not present or merged, SMEs are a good workbench to implement such integration.

Normalization Techniques

Starting from LEs' experience in the creation and maintenance of business processes, BONSAI suggests adopting normalized versions of relevant LEs' Business Processes.

In the BONSAI approach, normalized is intended as both simplified and optimized. Each process in SMEs has to be simple, easy to implement and maintain, because of the dynamic nature of SMEs. But also, LEs' business processes don't fit SMEs' needs because they require company structure and costs that a SME can hardly afford.

The first normalization technique is about reduction. If we consider a complex LE Business Process, and we try to fit it into SMEs' reality, most probably we need to apply such a technique. Reduction means that, where needed, a BBB should be eliminated from the business process. Let's take for example an enterprise process for the development of a product. Before the development starts, LEs usually spend effort and resources for prototyping. Since SMEs don't always have such resources, prototyping is cut off, and the development starts immediately after requirement specification.

The next normalization technique is about addition. It means that, where needed, a BBB should be introduced to the business process. Let's take for example a bug tracking system, containing all the defects detected on a software release in several phases of development: module test, integration, system test. Let's suppose that the Customer Support Department has access to the same bug tracking system, helping it to find the solution to end-user problems. The first remark is that the whole tracking system does not completely suit the Customer Support's needs, since it contains a lot of useless knowledge regarding system integration and module tests. Addition in this case covers the creation of a new BBB, responsible for filtering and isolating the defects found by the System Test, and currently not solved. In this case the Business Process has been optimized by the addition of a new process.

The next normalization technique is about workload decrease. In this case, to remove the bottleneck we identify the process causing it, and we change it. One or more blocks can optionally be added in order to mitigate change. Let's make an example. It has been detected in a software group that integration takes too much time compared to estimates. The reason has been identified in the lack of enough people to perform integration. The company is willing to hire new people, but it will take time, and the new people will be on board, properly skilled, far away from the release date.

Workload decrease can help in this way: we need first to decrease the Integration Team workload. We can accomplish this by changing the integration procedures: defects and change requests have to be affordably grouped, and integrated as a whole. In this way, the number of software builds decreases considerably. Of course, this approach can reduce control on the integrated code. In order to mitigate the risk, a code review activity is added right after development, with the aim of increasing code quality. In conclusion, integration has been simplified reducing the number of builds, and the process has been optimized introducing code review activity.

The next normalization technique, more intrusive than workload decrease, is about load rebalancing. Usually, when a bottleneck is identified in a process, there are two choices to solve it: add new people to increase the production level, or change the process itself. Workload rebalancing is all about process change: one or more processes are redefined in order to remove the bottleneck.

Let's make an example: a Core Development Team shrinks. The workload for each developer of the team increases, and as a consequence the Module Test Engineer workload decreases. To rebalance the workload, the Module Test Engineer, after completing the Module Test, is asked to proceed with the System Test right after the module integration. The Integration engineer, who has became the most experienced engineer on that module, can support the other engineers to understand how it is working. In conclusion,

workload rebalancing has impacted on the processes of the Core Development Team and the Module Test to fit the new structure of the CBB.

The next normalization technique is about project control. Software Project complexity can be measured with several parameters: costs, lines of codes, number of modules, number of people involved, etc. Small software projects are usually easier to develop than complex projects (additionally, a system composed by a single device is far less complex than a system composed of several interactive devices). The project control techniques aim to keep projects small, adopting a *divide et impera* approach: large projects can always be split in several smaller projects.

As a final consideration, we have seen that normalization can introduce risks. For this reason, a risk assessment and a mitigation plan should be considered when normalizing.

Synergy Improvement

Synergy is strictly dependent on communication. Synergy without communication is very hard to obtain. That's the reason why most of the best practices to increase synergy are focused on communication improvement. Communication improvements can be grouped in two categories, both complementary: tools and team attitude.

A lot of different tools can be used to improve communication in a software development group: versioning control, bug tracking system, shared project planning tools, wikis, forums, etc. All of them require first an effort to be selected, effort for customization, effort to teach to the team how to use it, and maintenance of the tools.

But tools are not enough, since we need people using them effectively. Actually any one of us make great use of a tool if we believe that can be useful. If not, it will be just another annoying task to be done. It should be noted also that "Individual and Interactions are more important than process and tools" (Agile Alliance 2001).

That's the reason why it is necessary to build up a communication and collaboration attitude around the tools. Such an attitude is not always easy to build, since it strictly depends on personal and cultural components. It is common understanding that a tool is welcomed when it helps people to do their jobs better and more quickly. So the tools, and the eventual customizations, should be focused on providing advantages to all the users involved.

The team attitude is also directly affected by project outcomes. To simplify, when the project is going badly, blame is on everyone, but when it is going fine, the merit should be upon everyone. It's especially important that rewards will be recognized by the enterprise itself, in order to keep personal motivation high. Meritocracy is the model that best helps to promote attitude.

A big communication problem is distance, especially when project teams are located far away. In this case, tools and proper project control make the difference. Also, as suggested by DeMarco and Hruschka et al. (2008), face-to-face contact is a good weapon to increase the team attitude.

Attitude for good communication is also strictly related to work conditions. When people are overloaded by day-to-day work, the time for communication is short. Communication requires time, and this time should be taken into account in each development plan.

Another relevant communication skill is the capability to share one's own knowledge. This can be a good accelerator in several situations. Let's suppose that a developer is making a hard piece of code. He has some doubts regarding implementation, or worse he does not know how to solve it. But he knows that other guys in the team have solved similar problems in the past, or they are masters of a programming language and surely they can give good advice. That sharing climate can help the developer to do a better job. It can be pretty obvious in a small team, working in the same office, but it is not so obvious when teams are located around the world. In this case the Project Manager has the role to spread around teams that sharing climate, assuring that this kind of code review activity will be brought off naturally by every engineer.

Consensus is also very important among the team. A widely used technique for improving consensus is Wideband Delphi (McConnel, 2006), used in the estimation process: each component of the team creates an estimate anonymously, then traces a plot representing the estimates on a timeline. A discussion follows on the average estimate. When decided, each component votes anonymously for approval or not: if any of the components don't approve, the discussion starts again and the process is repeated until a single-point estimate is approved. This technique, used especially in early-in-the-project estimates, has the side-effect increasing considerably the consensus among the team.

To improve synergy in the Core Development Team, the Project Manager has a very important role. As described by DeMarco and Hruschka et al. (2008), a project manager should be like a nanny, taking care that all engineers have the facilities needed to do their job, encouraging engineers to discuss new ideas and scheduling time for it, protecting the team like a wall from the rest of the organization, trusting them, supporting them and listening to them.

Adaptation

Following Conway's Law (1968), "the organization influences the structure of the code and the architecture". That's the reason why each Software Development Process should adhere to the organization's implicit and explicit Business Processes.

Implicit Business Processes are those processes that are not specified and documented, but they are executed and recognized by everyone. Most Business Processes in SMEs belongs to that category. Explicit Business Processes are those processes that are specified and documented. In both cases, people are taking responsibility for a part of the process. Let's take for example a company selling software for Android-powered devices, based on customer's detailed requirements. A customer is asking for a turnkey customized VoIP Phone software. Once the requirements are decided, an offer is produced by the company and when accepted, the development begins. In that case, it probably makes no sense to have an integration team, since the application is composed by just few components (VoIP stack, application) and requirements are fixed, since we are talking about a turnkey project. And probably it makes more sense to have a prototyping team, able to develop demos to show to potential customers.

Another example is about outsourced projects. Let's take for example a company developing telecom systems, and part of the system is outsourced to consulting companies. In this case, integration and short iterations are necessary to guarantee that the system is developed properly.
Another Adaptive aspect is about how processes are able to react to changes. We will cover two types of changes: estimation changes and requirement changes.

At the end of each iteration, the team improves its knowledge of the system under development. He knows how much it took to complete the iteration, if the initial estimate was good or not, which issues were faced, etc. All of this knowledge can be used to refine the initial estimate as suggested by McConnel (2006). McConnel's reasoning is based on Cone of Uncertainty: a plot where horizontal axis contains common project milestones, and vertical axis contains the degree of error found on estimates. The resulting plot is a cone, narrowing from left to right. This demonstrates how uncertainty is reduced milestone after milestone.

Additionally, at the end of each iteration the original requirements can change. This can be for several reasons: Marketing, watching at a software demo, realizing that something should be changed to fit the requirements better. Or the end-user asked for other high-priority features, or the market itself requires that a new important feature be present in the product. In any case, after new requirements have been discussed, a rescheduling is also needed, and a normalization process can be required additionally.

Conclusions

The BONSAI approach for the Software Development Process brings several advantages to the Enterprise. Since it is based on a single iterative approach, which involves most of the Enterprise, it can help to anticipate changes, before the product is completely developed giving a major competitive advantage to the Enterprise. The BBBs implemented in the processes can be reused and inherited (e.g. for hardware development, quality assurance, etc.), easing the implementation of further processes. The synergy between different teams is improved giving a competitive advantage to the Enterprise. Normalization techniques can also help to optimize an existing process especially processes coming from LEs: this allows advantage to be taken from LE's experiences. Adaptation techniques can be used for the definition of processes able to support continuous changes, these are very common in dynamic companies like SMEs. Costs are kept low, since no specialized engineers are required to adopt BONSAI. Risk assessment and mitigation plans can be used to understand the risk involved in each process change.

All of these advantages have the effect of improving the competition effectiveness of the Enterprise which can, with BONSAI, move the enterprise business towards a process-oriented and synergic approach.

References

Agile Alliance (2001). Manifesto for Agile Software Development. [Retrieved on September 7, 2010], www.agilealliance.org.

Ambler, S. W. (2002). Agile Modeling, Wiley Computer Publishing, New York.

Conway, M. E. (1968). “How do Committees Invent?,” *Datamation magazine*, April 1968.

DeMarco, T., Hruschka, P., Lister, T., McMenamin, S., Robertson, J. & Robertson, S., (2008). Adrenaline Junkies and Template Zombies, Dorset House Publishing, New York.

Di Leva, A. & Laguzzi, P. (2008). “I Business Building Blocks: dalla modellazione all’esecuzione del processo,” Proceedings of V Conference of the Italian Chapter of AIS (itAIS 2008), ISBN:978-88-6105-076-1, 13-14 December 2008, Paris, France

Ibrahim, A. B & Goodwin, J. R. (1986). “Perceived Causes of Success in Small Business,” *American Journal of Small Business*.

McConnel, S. (2006). Software Estimation. First edition. Microsoft Press.

Mills, H. (1971). “Chief Programmer Teams, Principles, and Procedures,” IBM Federal Systems Division Report FSC 71-5108.

Verbraeck, A. and Dahanayake, A. (2002). "Building Blocks for Effective Telematics Application Development and Evaluation," Delft University of Technology, Faculty of Technology