DEEP: DEveloper Engagement Platform - Balancing the dimensions in development environments

Gabriele Provinciali – gabriele@provinciali.com

Abstract

Beyond the methodologies and the software life-cycle management techniques, the human factor in a development team can be also represented in three main dimensions, such as integration (the level of technical coupling between developers and their tools), sharing (a common understanding and refinement of the best practices and experiences in software development) and affinity (the propagation of common value to enhance the developers' collective intelligence).

Human factor impact is heavily based on the concept of engagement: the paper will deepen the underlying implications of a common approach which allows to address integration, affinity and sharing using a developer engagement platform. The platform enables skills, experiences and – mostly important – ideas generation based on reward mechanisms and continuous proposition-refinement-execution of new ideas.

1. Introduction

The widespread adoption of lightweight methodologies in modern software development [1] has changed both the staffing processes related to the developers' organization and the way the developers communicate between each other.

The expected advantage in the agile practices lies in an overall better software quality (*Working software over Comprehensive documentation*) closer to the real customer needs (*Customer collaboration over Contract negotiation*) in an innovative fashion (*Individuals and interactions over Processes and tools*).

The advent and the spreading of agile methodologies and practices also changed the way a software project can fail, which includes the pressure to follow traditional practices and failure to trigger a cultural transition [2]. Barriers and concerns about agile adoption can be found a couple of recurring themes: loss of project control (lack of upfront planning, reduced predictability, lack of documentation) and decline of perceived trust in methodology (loss of management control, management and/or development team opposition to change).

The *Individuals and interactions over Processes and tools* statement also enforced the promise to take back the control of the software production directions and guidelines in the hands of the programmers: the generalization and the complexity induced by large development organizations is hardly applicable to small teams, often focused on a limited set of functionalities and frequent delivery schedules.

Developers' collaboration mechanisms converge, usually, in a single repository (*all-in-one*), which allows different activities – from group showcase and basic information storage to knowledge diffusion – and guarantees the simplicity and speed needed by an agile team.

What about developers' commitment? The leading causes of failed agile projects [2] seem to reside in the *lack of experience with agile methods*, coupled with *company philosophy or culture at odds with core agile values*. The "lack of experience" moniker doesn't necessarily relate to an incomplete training phase or to unneeded external pressures in developers' environments, and suggests researching of alternative forms of collaboration between community members, which is the scope of this paper. The developers' commitment is a key factor to motivate a transition from the usual information sharing routine to a more profitable practice involving generation of new ideas and methods to solve problems, with remarkable impacts on performance and quality [3].

1.1 Exploiting the Developers' Collective Intelligence

Transitioning from a dynamic multipurpose canvas, whose usage ranges from note taking to information sharing, to a *collective intelligence repository*, where the main scope is to stimulate, design, execute and measure new ideas to improve several aspects of the

software development lifecycle, requires more than instruments, tools & automation.

The idea generation phase requires developer's *engagement*. Tools and automation techniques, then, can be used to implement a system used by developers' to share and execute new ideas and monitor the progress of submitted ideas and projects.

A *collective intelligence repository* can be considered as a more ambitious project than an information repository essentially based on interactions. Within an information repository, developers can find useful and ready-to-go information about code techniques and solved problems, pending issues and best practices. A collective intelligence repository extends this approach to include a number of innovations, including but not limited to:

- *Shared Artifacts Generation*: the developers can propose new ideas/projects, and seek for sponsorship and co-authoring.
- *A Developers' Economy*: successful implementations are rewarded with a virtual currency (Developer Units - DUs), subsequently exchangeable in different raw goodies (e.g. learning credits, development time).
- *New Roles for Community Members*: new ideas and their execution can be shared amongst individuals willing to contribute and act as a proposer, a refiner, a supervisor, etc.

The whole system is based on a triple made of engagement, participation and incentives, and simultaneously targeting the developers' community as a whole as well as the single participant to new initiatives.

2. Wiki and Beyond

As we've defined the *dynamic multipurpose canvas* technologies earlier, the wiki technology is a peculiar 'all-in-one' approach used by developers to share information. The wiki adoption rate in different industries has been remarkably wide and well over the restricted circle of software development, thanks to factors that include the ability to provide content/document management capabilities and project management information hosting.

The advantages of a wiki-based approach in software development can be detected in three areas, such as *immediacy* (ease of use, WYSIWYG, user-serviceable data structures), *offloading* (e-mail, traditional document management interactions) and *flexibility* (ability to accommodate different functions within the same container). The combination of these qualities makes the wiki a powerful instrument [4] to boost productivity and accelerate the organization and sharing of developers' generated content.

On the engagement side, the wiki context has a limited set of features that allow interactions between peers within a given activity to mediate different takes and opinions on a particular subject, and insufficient abilities to engage the developers in generating new ideas and projects.

More generally, while the wiki technology is good at content creation and sharing via social features (build consensus and diffusion by means of comments and *likes*), it might not completely relate to developers' engagement and commitment to create and facilitate substantial *improvement*, which may be outlined as the main scope of the collective intelligence repository, following the Kaizen industrial orientation [5].

The success of the wiki concept implementation has spawned newer generations of public repositories such as Stack Overflow and Big Resource, focused on programming languages support and interaction with developers.

3. Stack Overflow – User Experience and Business Model

As of March 2015, the Stack Overflow website (http://www.stackoverflow.com) has proven to be one of the most successful knowledge sharing platforms based on user-generated content on a plethora of programming languages (Java, C#, Swift, Python). Created in 2008, the purpose of the site has been declared as a free alternative to other commercial information exchange platforms, and – at present – the number of interactions between developers ranges in the tens of millions.

The innovation behind Stack Overflow initiative can be found in focusing on a single expertise area (specific queries on programming languages and techniques) and the active participation of the members by means of questions & answers interactions and earned reputation within the community.

3.1 Shortcomings

Being based on Q&A and social mechanisms such as reputation, the Stack Overflow model presents good suggestions and ideas to get started, although its implementation is not completely applicable to organizations with peculiar operating environments.

The idea generation phase is still not completely addressed, since the inner mechanisms provided by the Stack Overflow model are not suitable to provide a continuous interaction rate between developers, thus preventing an acceptable engagement level. Few excellent technical answers that are massively voted from community members could sustain a developer reputation for a prolonged period of time, even if interaction between the developer and the community is intermittent or low.

4. Engagement Via an Innovation Platform

The definition of developer engagement, in terms of relationship with peers, can be seen in two main areas: the absorption of the group culture and values to boost the individual productivity (*inbound engagement*) and the capability to devise new ideas to improve the collective intelligence (*outbound engagement*).

The platforms and the interaction models examined in the previous section – based either on sharing and collaboration or reputation score – are mainly centered to amplify individual productivity via inbound engagement: some aspects covered by this paper underline the importance of the outbound engagement and the related tools to increase the ability of developers to produce innovation.

The human factor (and its impact) in a developers group can be located in three main dimensions, such as *integration* (between developers and their tools), *sharing* (the best practices and experiences in software development) and *affinity* (value to enhance the developers' collective intelligence).

Some examples can be made upon useful artifacts that can be designed, developed and made available within these dimensions, conveniently stimulated by an engagement platform that catalyzes developers' interaction and teamwork. The *integration* dimension could accommodate ideas and artifacts such as plugins for development tools and IDEs, test utilities, libraries and other reusable software. The *sharing* dimension is mainly about experience, so it could host How-Tos and valuable timesaving information. Finally, the *affinity* dimension is devoted to house artifacts related to continuous improvement of individuals and groups.

Dimensions can be considered as the foundation of a threefold repository, where each one is a target for new ideas, aimed to augment the innovation level, and that can be correlated with other technical elements (*software architecture*) and participation incentives (*social triggers*) to outline a blueprint for an innovation platform where the developer – at last – can be put again in the spotlight.

4.1 Technical Elements

A platform capable to address outbound engagement in the integration, sharing and affinity areas should contain – at least – three technical segments: a subsystem to regulate and mediate interaction between developers, a subsystem to manage a potentially long-running process related to idea refinement, and a container of the developer-generated ideas and information, as in the following picture, which represents the DEEP (Developer and Engineering Engagement Platform) environment building blocks.

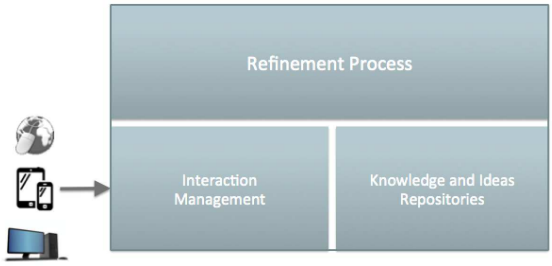

Figure 1. DEEP building blocks

The three subsystems can be based on available technologies in modern middleware implementations such as *UX technologies* (interaction), *business process management* (refinement) and *content management* (knowledge and artifacts). The communication between components can rely on standard transports and protocols (REST, Web Services) and propagation of credentials (identities and roles).

While no other technical dependency is defined at this stage, the ease of implementation can be reasonably related to the maturity of the single subsystems technologies (vertical maturity) and the maturity of integration between subsystems (horizontal maturity).

The platform must be seen as a facilitator and not as a hurdle, thus the design, implementation and rollout phases should be aligned to the lean software development principles.

4.2 Interaction

The interaction management subsystem is responsible to handle communications between actors (developers, managers) and present in a suitable form the information, mediated by the Process layer, located in the Knowledge repository. This layer should be provided as an independent contrivance capable to

consume and produce information in a decoupled fashion from the physical device where the interaction is requested.

The interaction management subsystems should be also responsible to store the profiles and the incentive position and ranking related to each user. It is strongly suggested that all the required UI should be compliant to the RWD (Responsive Web Design) principles [7], in order to facilitate interaction between parties (ideas for improvement and innovation can happen anytime) and remove the obstacles linked to usage of a desktop/notebook PC system.

The UI should be maintained lean and streamlined as the 'social' aspects are implicit and inherent to the main platform goals, and the incentive system is not necessarily connected to the usual social networks primitives (i.e. no stringent need for "Likes" or "Shares" actions).

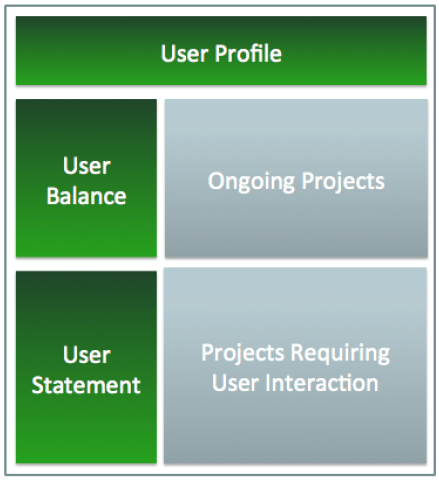

Figure 2. Sample UI in the interaction management subsystem

Logging in, the user could be presented with a number of sections, such as:

- A list of ongoing projects within the DEEP platform potentially selectable for evaluation and eventual sponsorship – a viewer of activities sortable by funding, date or other parameters.
- A list of activities requiring user attention or interaction – the user dashboard.
- The user statement and balance, reporting information about the position related to incentives.

The section centered around the user position, state and identity are directly retrieved from the interaction management container (in green), and the section centered around contents, ideas and projects are retrieved, mediated by the process layer, from the knowledge and content repository (in grey).

4.3 Process

The process layer is the appointed container to run the refinement process for submitted ideas and projects, whose main goal is to transform the idea in an executable item.

Since the refinement activities are performed by a plethora of roles, and each role has different capabilities and visibility on the activity stream, the process is defined as a long-running transaction, with several interactions between parties involved. The roles breakdown, list as a pure reference example, could be done as the following:

- **Proposer**: the individual initiating the innovation process by submitting an idea pertaining to one or more dimensions. The proposer should be able to describe the idea with an acceptable detail level in order to be published on the Ongoing Projects Dashboards and gain attention to seek funding within the community. The Proposer individual can also impersonate the Deliverer.
- **Refiner**: the individual that helps the proposer by being his counterpoint. Once an idea has gained attention and funding, the interactions between the Proposer and the Refiner are iterative until the idea is ready to be enabled.
- **Enabler**: the individual that supervises the interwork between the Proposer and the Refiner and facilitate the production of the necessary artifacts to be submitted for the approval.
- **Decision-Maker**: the individual that ultimately enables the transformation of the idea into a project by analyzing the profitability and the impact on developers' groups.
- **Deliverer**: the individual that performs the execution of the project after the idea has been given a GO from the Decision-Maker.

Other roles can – of course – be implemented for more sophisticated idea generation flow, which will depend on technical and organizational attributes to be taken into account for each developers group.

The container should provide allowance to such interactions in an easy and intuitive manner, taking advantage of automation features and flexible design

instruments. The diagram below depicts a sample flow with basic interactions between the mentioned roles.

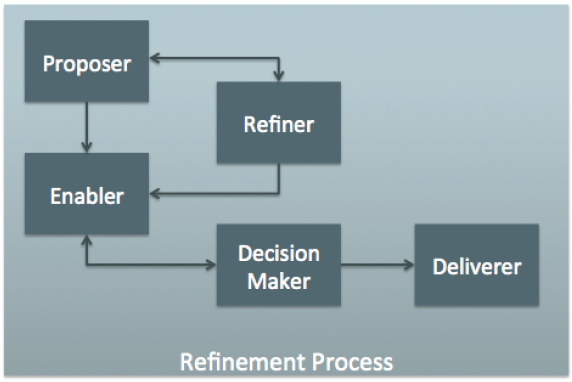

Figure 3. Sample roles in the process layer

The process container should be able to translate the interaction between parties in an executable flow, updating both the interaction management subsystem and the knowledge repository with up-to-date status and contents for a given project or idea.

4.4 Knowledge and content repository

The idea generation phase will be able to produce unstructured contents that might take many forms, such as code snippets, images, textual documents, and other digital asset formats.

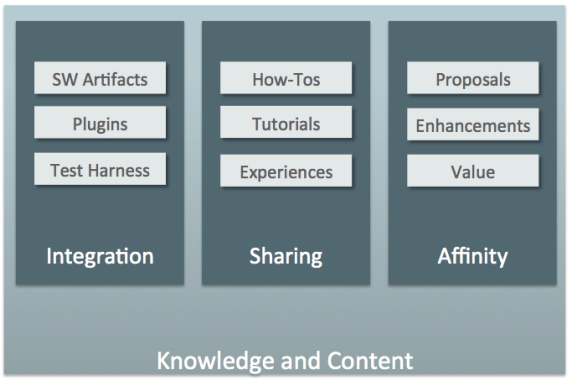

Figure 4. Dimensions in the threefold repository

The designated container that will lodge the assets produced during the idea generation and refinement phases will provide content metadata handling for practical search, retrieval and reference activities, and will be shaped upon the dimensions (integration, sharing and affinity) described in the earlier sections.

5. Incentive Mechanisms

Stimulating developers to be a significant part of an innovation program requires more than sharing and reputation ranking: it requires *incentives*.

Commercial enterprises can easily associate the incentives with accelerators that revolve around financial instruments (bonuses, shares). The developers' community could instead take advantage of an economy based on a virtual currency, conceivable as *Developer Units* (DU). DUs are the vector by which developers can submit and actualize ideas, get acknowledgement/credits and improve efficiency of software development.

DUs can be freely and wisely spent to sponsor new ideas and projects and accrued working as a main actor on projects, enabling the conversion of the ideas into executable projects that can represent a tangible benefit for the Organization.

5.1 Creating an economy based on innovation

Every developer that enrolls to the platform can be equipped with an amount of DUs, which can be homogenous for all developers – if compatible with the organizational environment – or diversified, based on other factors.

The submitter of an innovative idea connected to one of more developer dimensions can demand DUs (thus attention, resources and visibility) to *crowdfund* the necessary work to convert the idea into a project with the help of other individuals/roles.

The various roles involved in a successful implementation of an innovative idea – or better, the virtual team involved in the process – might be compensated, in an adequate and weighted fashion, with the DUs accrued during the crowdfunding phase.

The whole scope of the incentive mechanisms is to stimulate *attention*, *improvement* and *innovation*, as well as creating a developers' economy based on idea generation, artifacts quality and teamwork.

5.2 DUs Conversion

DUs accrual would be an integral part of the engagement platform economy, as one the mechanisms provided to boost participation. The DU concept, however, is not restricted to generation and maintenance of community-related acknowledgements and credits, or as a sole reputation gauge: a virtual currency can be considered also as a means to gain professional advantage (self-improvement) and

practical help during real-world project executions. Therefore, DUs could be designed as convertible goods.

The conversion of DUs in other assets – compatible with the organizational structure hosting the platform – represents an added value to encourage developers and developers' group participation. Possible examples of converted assets are the following:

- **Time Bank**: accrued DUs can be converted in *time units*, such as available hours to be reclaimed from other developers enrolled to the platform and usable to produce/review code, documentation, testing activities.
- **Learning Credits**: accrued DUs can be converted in *learning credits*, such as vouchers to attend training courses, access workshops, and acquire technical books and documentation.

6. Conclusions

The current generation of sharing and collaboration platforms, available in different flavors, and often exhibiting a rich set of specific features, are mainly focused to enhance the user experience or dedicated to enable personal motivation by means of social interaction or reputation measurement and management. The aim of such platforms is to mitigate or cut down the *information friction*, enabling knowledge to be shared amongst participants.

The DEEP platform exposes other significant areas that a developer engagement platform needs to address to become attractive, such as:

- *Incubation*: facilitation in creating new ideas and projects and help making them feasible with other professionals involved.
- *Focus on matter*: the proximity of new ideas to the three dimensions centered on integration, sharing and affinity, empowering the developer job.
- *Convertible Rewards*: tangible benefits for developers who actively participate and make innovation available.

The DEEP platform is a proposal – independent from the underlying technology – that aims to put the developer back *at the center* of the innovation process by rewarding the idea, the individual, the initiative and the group effort.

References

[1] Agile Manifesto (www.agilemanifesto.org)

[2] State of Agile Development Survey Results, 2009 (www.versionone.com/state_of_agile_development_survey/09/page5.asp)

[3] Ostroff, C. (1992). The relationship between satisfaction, attitudes and performance: An organizational level analysis. Journal of Applied Psychology, 77, 963-974.

[4] Andersen, E. (2005). Using Wikis in a Corporate Context. Norwegian School of Management.

[5] Tozawa, Bunji (1995). The improvement engine: creativity & innovation through employee involvement: the Kaizen teian systems. Productivity Press.

[6] Atwood, Jeff (2008). (http://blog.codinghorror.com/introducing-stackoverflow-com/)

[7] Marcotte, Ethan (2010). (http://alistapart.com/article/responsive-web-design)