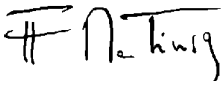
METHODS & TOOLS

Practical knowledge for the software developer, tester and project manager **ISSN 1661-402X**

Summer 2014 (Volume 22 - number 2) www.methodsandtools.com

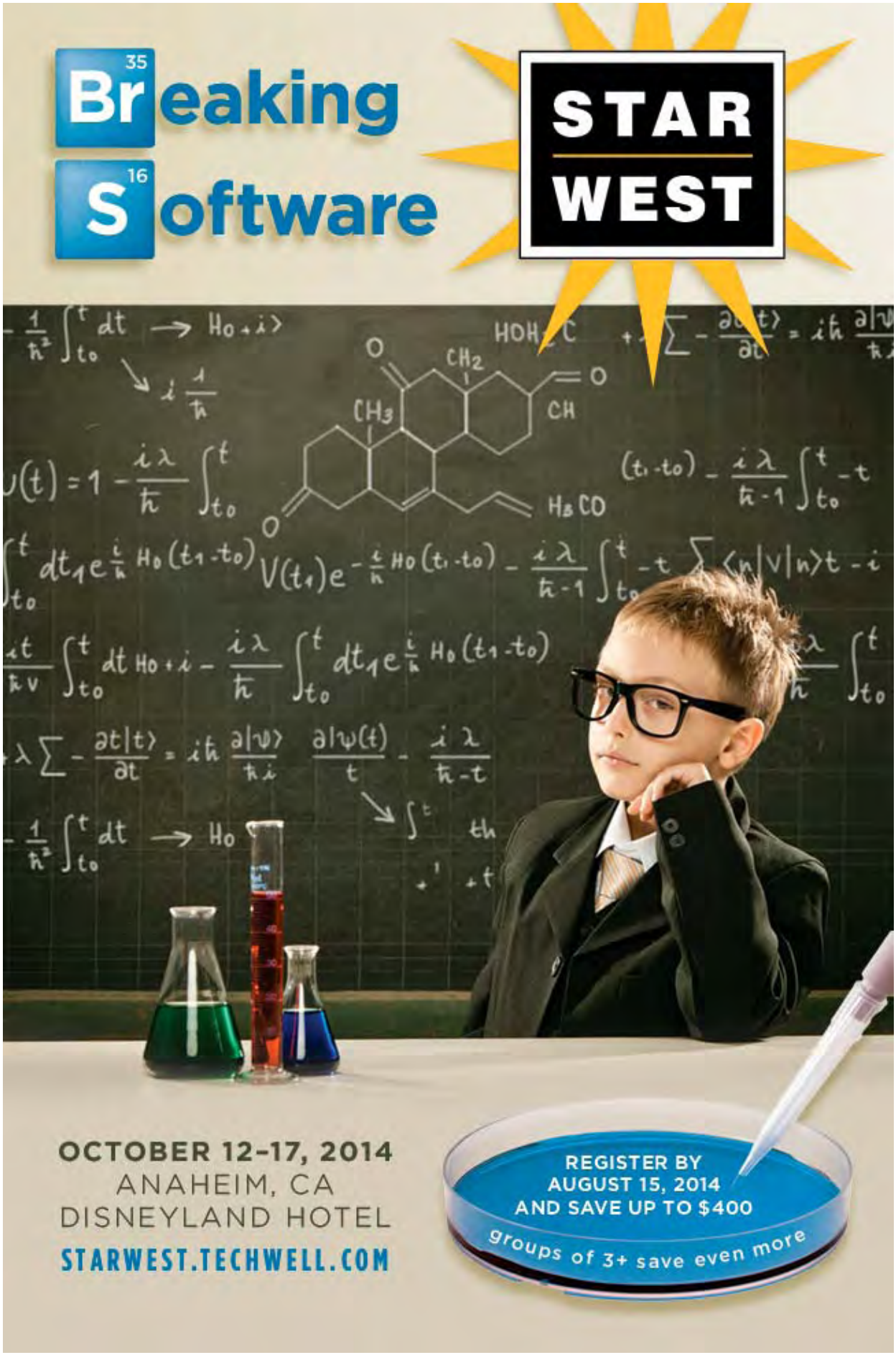
Is Agile Dead or Can Good Software Development Scale ?

As Agile becomes a widely accepted software development approach, many large organizations have adopted it. If the topic of scaling Agile has been discussed for many years, new proprietary approaches have recently emerged to achieve this goal. At the same time, I have also read an increased number of opinions about the fact that Agile is not working or might be dying. The notion of scaling has two dimension. An approach should be able to be used in larger projects and it should be also used by different software development organisations, each having their own culture. In a simplistic view, there are two approaches to craft a software development organization that produce quality software to meet the needs of users and produce value for business. You can define a prescriptive process that institutionalize "best practices" for software development and you implement them through training and a monitoring system to ensure that people follow them. On the other side, you can instill a continuous improvement movement to your (good) software development people and you let them gradually create their "good software" process. This could be summarized as CMMI/RUP versus Agile in a simplified view. Whatever approach you take, you need discipline and at the end it is only people making decisions. We might think that we are all above average software developers or project managers, but statistics say the contrary. I doubt that the average quality of developers, project managers and business users has improved in recent years. It is then inevitable that trying to scale any approach has its limits and increases its chances of failure. You should not consider your favourite approach as a silver bullet, but, paraphrasing Churchill's definition of democracy, as the worst approach to software development except all the others that have been tried. As a conclusion, I will cite what Graig Larman and Bas Vodde wrote in the first page of their first book "After working for some years in the domain of large, multisite, and offshore development, we have distilled our experience and advice down to the following: Don't do it". You can try to scale everything, but you have to be aware of the negative side effects that are associated to this journey. So just follow the advice from the same authors "start small with a group of smart people and only grew when it really start to hurt".



Inside

STARWEST Conference - Click on ad to reach advertiser web site



Kanban for Skeptics

Nick Oostvogels, http://www.skycoach.be, @NickOostvogels

As a change agent, you constantly need to reassure people that the path we follow is worthwhile traveling. This need is often expressed in the form of critique and difficult questions. When I coach teams, this is often the case. The same thing happens when introducing Kanban.

However, I noticed that Kanban raises much harder questions on a management and leadership level, once people are introduced to the basics and start to explore the subject on their own. The type of questions Kanban raises seem to be hard to answer without lapsing into hour-long discussions. This is normal because Kanban is a change management approach, not a methodology and therefore much less prescriptive. In order to provide reassurance, as a coach, you need to trace the questions all the way back to the principles of Kanban, which are grounded in Lean thinking [1].

This is why I wrote a free e-book 'Kanban for Skeptics' [2], that lists the 5 most common arguments against Kanban.

What is Kanban?

When the term Kanban is used in the field of software engineering, people think it is only the practice of adding Work In Progress (WIP) limits to the different stages in a development lifecycle.

In theory, Kanban is a change management approach that employs a WIP limited pull system.

The limited set of rules and principles help people focus on customer value and avoid stacking up half finished work, which is still a common issue in the software industry.

David Anderson has written a nice summary on the principles of Kanban [3]:

First follow the foundational principles:

- Start with what you do now
- Agree to pursue incremental, evolutionary change
- Initially, respect current roles, responsibilities & job titles

Then adopt the core practices:

1. Visualize workflow
2. Limit Work In Progress
3. Manage Flow
4. Make Process Policies Explicit
5. Improve Collaboratively

But because Kanban is not a process, people will perceive the change differently. In day-to-day life, the way they work won't change much, in the beginning. The only things they will feel in their day to day job are the principles that employ a WIP limited pull system. These will slowly introduce different behavior and continuous improvement.

WebPerformance Load Testing Software & Services - Click on ad to reach advertiser web site

Because your web site should be in better shape than you are.

Training the NYC Marathon live website to be the fastest in the world for three years running.

New York Road Runners is the organization behind the world-famous New York City Marathon, which sends 40,000 runners across the city's five boroughs for 26.2 miles each November. Fans register to follow their favorite runners online, and these individual status pages are updated continuously until the exhausted and exhilarated runners cross the finish line, 20-30 participants per second.

For the past three years Web Performance has load tested the servers that provide live race information. We test and tune the capacity of the Athlete Tracking Website up to a level of 100,000 concurrent users, and the site handles the live race traffic without breaking a sweat.

Load Testing Software & Services

webperformance.COM

5 common arguments

In my e-book Kanban for Skeptics, I explore 5 common arguments against Kanban.

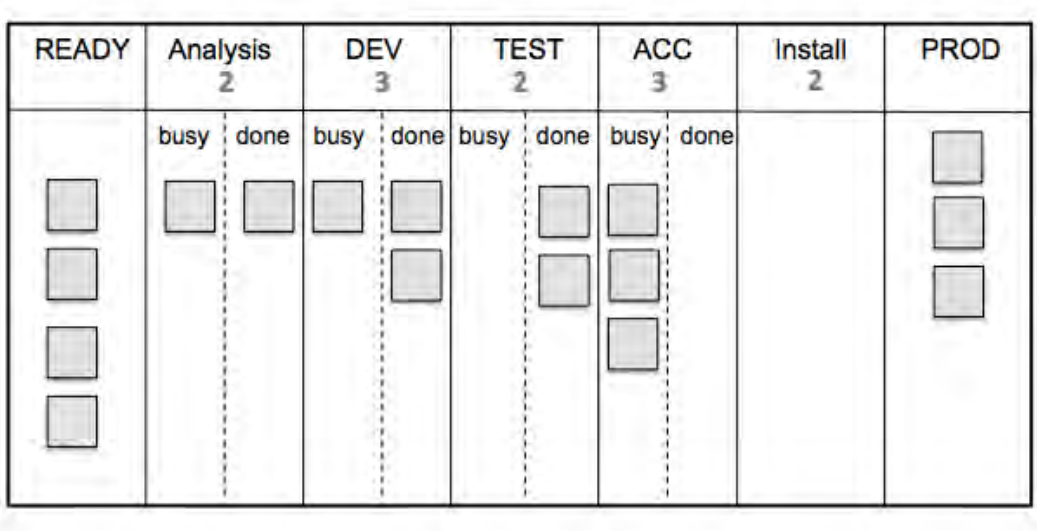
- We lose our ability to plan.
- It will take longer.
- Things will get stuck, we can't keep WIP limits.
- Stakeholders don't care about feeding the flow.
- We will lose team cohesion.

Although these arguments are invalid, we must acknowledge them as challenges for all leaders trying to introduce Kanban in organizations. My e-book will help you get the discussion started.

Let's explore one of the arguments.

Things will get stuck, we can't keep WIP limits

Work In Progress (WIP) limits visualize the maximum capacity of each stage in the value stream. If a WIP limit of 3 is set on the development column of the Kanban board, this means the developers can't work on more than 3 tasks simultaneously.



When people hear about the concept of Work In Progress limits, their first reaction is that it will never work in their organization. They expect a major efficiency drop if they would apply it to the different workflow stages. "Our testers can never keep up with the pace of our developers when testing the features. Developers would be idle for half of the time". This is applicable to all different roles in the organization all the way up from sales to operations downstream.

The way most organizations fix this is by working asynchronously. When testers are looking at the features, development is starting to work on new requests, to maximize their efficiency.

SpiraTeam Complete Agile ALM Suite - Click on ad to reach advertiser web site

Are You Tired of Having Separate Tools for Requirements, Testing, Bug-Tracking and Planning?

It's time to try a better way.

spiraTeam®

The most complete yet affordable Agile ALM suite on the market today.

Learn more at: inflectra.com/spiraTeam

www.inflectra.com
sales@inflectra.com
+1 202-558-6885

inflectra®

Remember that Kanban doesn't focus on maximizing resource utilization. Instead it focuses on maximizing end-to-end flow efficiency, which is a completely different goal.

Improving flow

The focus of Kanban on end-to-end efficiency all points towards pulling value from the customer with as much added value as possible. So instead of maximizing utilization rates, the team focuses on getting value through the flow as quickly as possible. In some cases this may cause people to be idle, making sure they are not working on things that cause an increase in work in progress and the risk of a value mismatch.

Instead of all blindly starting to produce as much as possible, the team focuses on improving the flow by removing bottlenecks [4] and redesigning the flow in order to keep up with the new stream of demand. This is an exercise that never stops, due to the variability in the consumer market. Thanks to the clear focus on end-to-end flow, the team makes sure they keep working in a pull system, driven by customer demand.

If we're honest, we all know that it's in our nature to start looking for new challenges when the major problem solving of a task is done. This leads to a huge pile of tasks that are only 90% finished. The end-to-end focus of Kanban will push you to finish the remaining 10%.

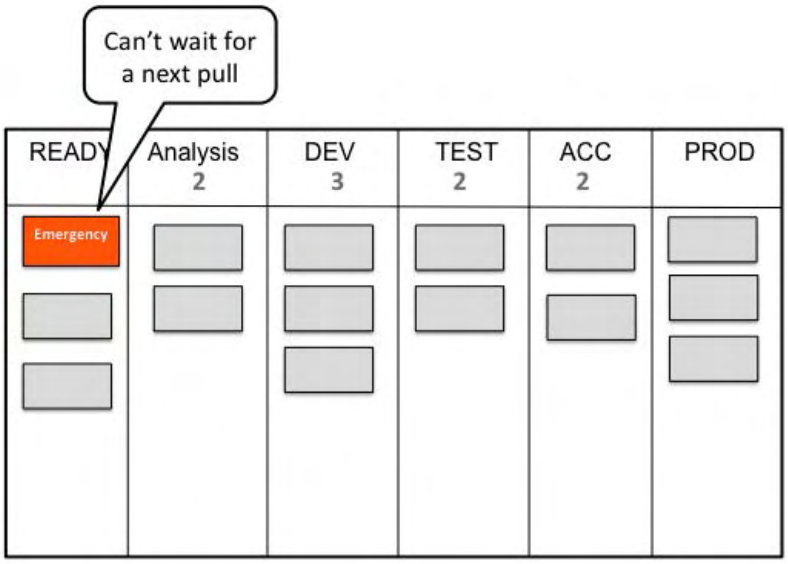
There is no exact science on determining WIP limits. As soon as they are chosen, bottlenecks will appear. This doesn't have to be a disaster. When the organization embraces the idea of a pull system, they will understand that WIP limits help to improve end-to-end efficiency. The first signs of a bottleneck appearing will drive continuous improvement. People will start to get nervous because they can't pull a new feature to work on.

It will drive team members to collaborate on 2 levels. A first reaction will be to go out and check what the problem is. Why are you stuck? Can I help you to get going again? This will eventually get the flow going again, but soon things will get stuck again, leading to the second reaction. How can we solve this bottleneck? If you think about it, the WIP limits cause the team to self adjust to changes that affect the end-to-end flow.

Emergencies

Another common remark is "We're not able to keep WIP limits because of emergencies that pop up from time to time". In a well-balanced flow, all stages in the Kanban flow are close to their WIP limits. This means that when an emergency pops up, the team won't be able to start solving it until new capacity becomes available. This is not acceptable because an emergency can't wait for half a day before someone starts working on it.

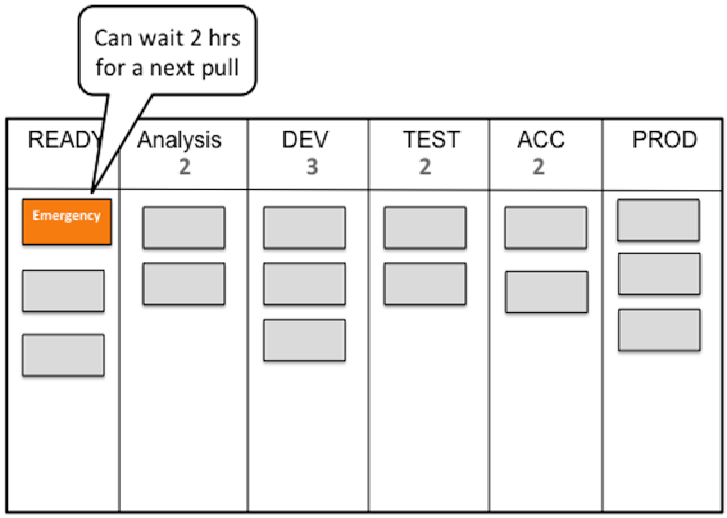
Visual Management with TargetProcess - Click on ad to reach advertiser web site



Reality proves to be different, even in Kanban land. When an emergency arrives, it needs to be dealt with, immediately! This means WIP limits can be broken. How- ever, the decision needs to be made very consciously because of the impact.

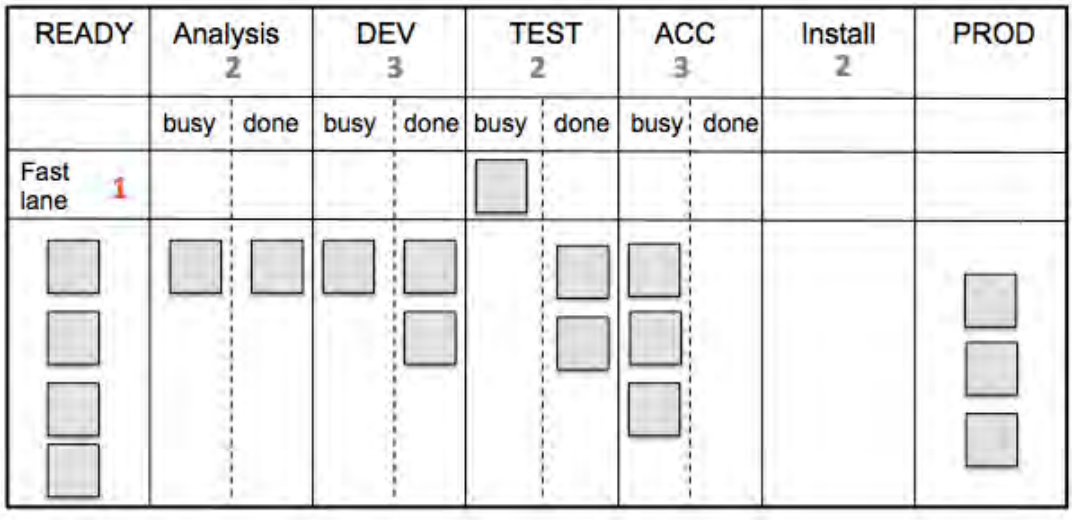
Introducing an emergency into the flow causes people to drop what they're doing and switch context. This comes with a cost. The flow will be interrupted and lead times will be impacted. You probably understand that it's important to reduce these interruptions to a minimum. Otherwise variability of the Kanban flow will increase, tearing down its reliability and frequent delivery of value. It doesn't take long before you fall back into estimating and planning.

One way to reduce the number of emergencies that interrupt the flow, is by making sure new items are pulled from the queue on a regular basis. When on average, it takes only 2 hours for a feature at the top of the queue to get pulled, far fewer emergencies need to be pushed into the flow. When you can rely on the pull frequency of the Kanban flow, it's much easier to just prioritize it in the queue instead of interrupting everybody's work.



Off course there will always be emergencies that need to be dealt with immediately. This is where the concept of Classes of Service [5] can offer you some flexibility. By categorizing the type of work that enters the Kanban flow, we agree to the service levels that apply to these different categories. Some categories have much higher due date performance than others. This gives you some room to absorb variation because of emergencies that disrupt the flow.

Many Kanban implementations use a fast lane to deal with emergencies. This is a horizontal lane that spreads across all workflow stages and has a WIP limit of it's own.



When an emergency arrives, for instance the payment module is offline, it is pulled through the fast lane with priority over all other work items. This is often the case when you work on an existing product and critical issues can appear at any given time. At that point, the WIP limits will obstruct the hotfix that needs to be released as fast as possible.

By using a fast lane, we implicitly give an emergency priority over all other work in the flow. But to prevent the fast lane to disrupt the flow too often, we will also limit its work in progress, only 1 item can reside in the entire horizontal lane across all workflow stages.

Summary

Off course introducing WIP limits is going to cause problems! The question is, why is that bothering you? The purpose of WIP limits is to make it easier to work in a pull system that is more effective from start to end. The WIP limits will signal an uneven balance in your flow. From that point, you can start working on improving it.

Free e-book

You can read about the other 4 arguments against Kanban by downloading my e-book for free at leanpub.com/kanbanforskeptics and automatically get next versions in the future.

References

1. Womack, James P. & Jones, Daniel T. Lean Thinking: Banish Waste and Create Wealth in Your Corporation. NY: Free Press, 2003.
2. http://leanpub.com/kanbanforskeptics
3. http://www.djaa.com/principles-kanban-method
4. Goldratt, Eliyahu M. & Cox, Jeff. The Goal: A Process of Ongoing Improvement. Boston, MA: North River Press, 2004.
5. http://www.dennisstevens.com/2010/06/14/kanban-what-are-classes-of-service-and-why-should-you-care/

OnTime Scrum Project Management Tool - Click on ad to reach advertiser web site

Using a Model To Evaluate and Improve Mobile User Experience

Philip Lew, XBOSoft, http://www.xbosoft.com/

User Experience (UX), a term that first appeared a decade ago, has become a sought after aspect of quality in most modern day products. However, there still lacks a formal definition or standard for UX not only in contemporary research but also in common practice. Some relate it with core usability aspects such as efficiency and effectiveness while others describe it in terms of subjective and hedonic user needs or feelings such as user satisfaction and pleasure. Yet others call it the 'wow factor'.

With advent of smartphones and other mobile devices (such as tablets), users have quickly shifted from conventional desktops to highly sophisticated mobile and now wearable devices. Their enormous advantage of on-the-fly connectivity via Internet and flexibility has lead to rapid proliferation of software applications on this platform. Switching from desktops to the mobile platform has been a challenge as user expectations are high and competition is fierce. Competition stems from two key elements. First, the subscription economy where there is little hesitation to switch to a similar competing application. Second, the mobile platform and app store distribution has significantly decreased barriers to entry such that a garage-based developer can develop and distribute an application on a mobile platform at a fraction of the cost of previous methods and channels. This makes usability and UX a key factor in application uptake. In other words, downloads and success.

Now, with the domination of smartphones, these "smart" and sophisticated devices have many UX challenges different from conventional applications (deployed/installed on desktops). For example, mobile smartphones usability is constrained by small sized screens (with low resolution) and keypads. With mobile applications, UX becomes even more complex due to the mobile's natural characteristics, some of which include:

- Wide audience: Smartphones are getting popular day by day and it is not uncommon for some people to have 2 or more smartphones. From desktop to web-based was one level of user diversity. Now with mobile, we go up another level in user diversity with a user group that has expanded way beyond traditional 'computer users'.
- Environmental conditions: The environment in which mobile devices are used is constantly in flux and mobile devices and their applications must adapt and change behavior according to environmental conditions. For example, depending upon the amount of light, the screen brightness or contrast can adjust automatically.
- Personal association: Mobile smartphones are not just a piece of electronic gear. Rather, users take them everywhere they go (yes, even in the bathroom) and they quickly transform into a personal belonging after being taken out of the box. This is much different than a traditional desktop that stays in one place. The transformation into a personal belonging is due to personalization of the device that includes settings, data, and software applications, not to mention other personalization aspects such as covers and earphones.

Due to all of these factors, it is not surprising that as mobile applications inundate the app stores, Mobile User eXperience (MUX) is starting to garner serious attention because some app developers are wondering why their app is not getting much uptake while others who may have less features for more money are. If you consider that if a user cannot learn your mobile app in 30 seconds, they will most likely uninstall it and find another, you better take MUX seriously.

Ranorex Automated Testing Tool- Click on ad to reach advertiser web site

Some of the key factors or characteristics that should be and are mostly considered when designing and evaluating MUX include:

- Aesthetics
- Efficiency and effectiveness
- User customization

Most of these factors focus on how the phone and its software operate, how the buttons are placed and operate, and how easy it is for users to complete their everyday tasks. However, systematical improvement of MUX requires a model-based approach that defines categories or characteristics for evaluation along with attributes. In this way, different software, or versions of the same software can be evaluated in a consistent way. When it comes to modeling usability and UX, most research points to ISO 25010 as a starting point. ISO 25010 sets forth a generic software quality model that includes such product characteristics as operability, reliability, performance combined with in-use characteristics as usability, efficiency, and effectiveness. The ISO 25010 is designed to be flexible and can be modified depending on the context, in our case, in evaluating mobile application quality.

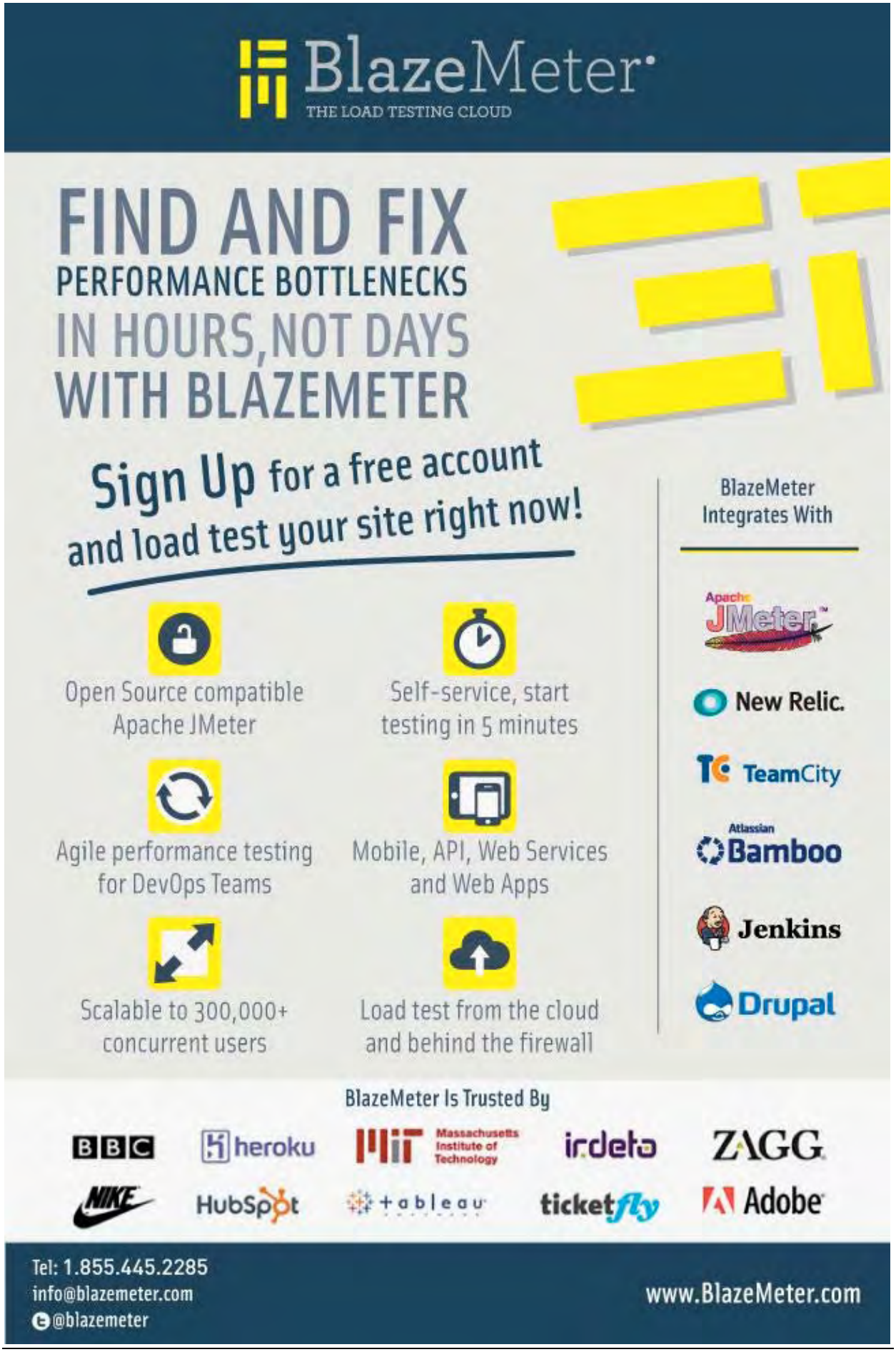
Developing a Model to Evaluate and Improve Mobile User Experience

Starting with the ISO 25010 quality model, we don't need to start from scratch, but rather extract parts of it to use as a basis for evaluating and improving user experience for the mobile paradigm. Before we dive into a model, let's pose some questions that a model should help us to answer and solve:

- What do "horrible usability" and "better UX" mean?
- What is the relationship between Usability and UX? Are they synonyms?
- Evaluating the success rate of users completing tasks correctly. Is this the same as or directly related to UX?
- If users are highly effective in completing tasks but they mostly get unsatisfied, then does UX score still high?
- Does UX depend on application Usability only or also from other characteristics such as Functional and Information Quality, Security, Reliability, and Efficiency?
- Is UX a quality characteristic of the software itself or of an application in use? And what about Usability?

ISO 25010 [1] provides a good starting point and as put forth in the standard, it is designed to be flexible depending on the needs and context of the organization. First, let's cover some concepts and definitions of the model that we can apply to our efforts in modeling mobile user experience. ISO 25010 outlines a flexible model with product/system quality, also known as internal and external quality (EQ), and system-in-use quality, also referred to as quality in use (QinU). Product quality refers to attributes that can be evaluated with the app in execution state both in testing and in operative stages; while QinU quality refers to what can be evaluated by end users when actually executing tasks in a real context of use. As mentioned earlier in the last article, the mobile app context of use is one of the primary reasons that UX is harder to evaluate and achieve. So, QinU deserves special attention for mobile applications.

Performance Testing with BlazeMeter - Click on ad to reach advertiser web site



ISO 25010 also delineates a relationship whereby product quality 'influences' QinU and QinU 'depends on' product quality. This makes intuitive for instance if the buttons are placed in certain locations (a product attribute), this will 'influence' a user's productivity and experience if they can't find the buttons. Furthermore, usability is a product quality characteristic, while Effectiveness, Efficiency and Satisfaction are QinU characteristics (see figure 1).

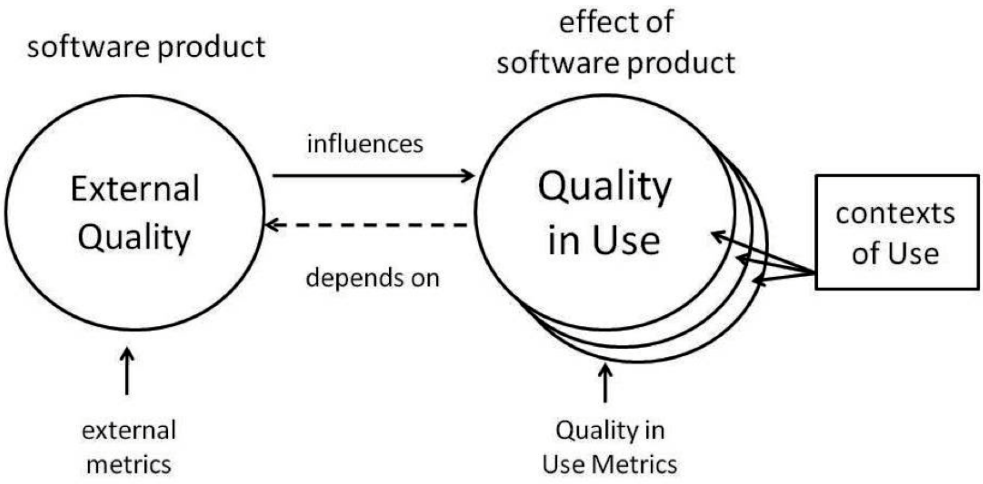


Figure 1. ISO 25010 quality views

However, Actual Usability and User Experience are missing concepts in ISO 25010 standard. To bridge this gap, in previous research, we used the ISO 25010 as a starting point, and developed 2Q2U (Quality, Quality in use, actual Usability and User experience) v2.0 [2 and 3], which ties together all of these quality concepts by relating product quality characteristics with Actual Usability (part of QinU rather than Product quality) and UX as experienced by the end user.

The purpose of the 2Q2U quality framework is to instantiate the quality characteristics to evaluate and conduct a systematic evaluation using the 'depends' and 'influences' relationships. Relevant features of mobile apps with regard to Usability and UX in the light of 2Q2U v2.0 quality models [4] include:

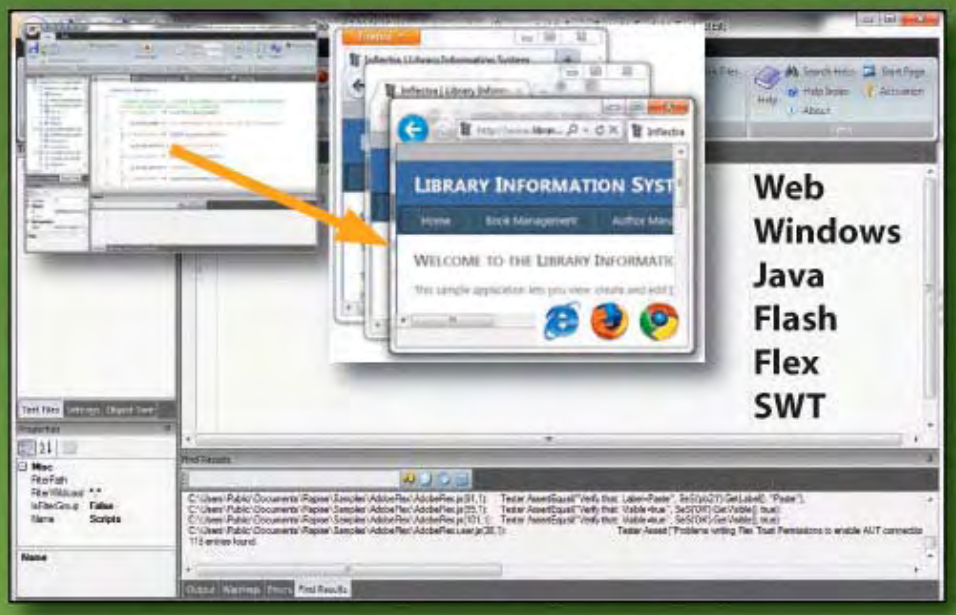
- Typing/input: which includes search bars, and other data entry fields whereby the users should be assisted as much as possible to reduce errors and the 'cost' of typing. This includes such measurable attributes as default values, default value removal and shortcuts.
- Entry widgets such as carousels, drop down boxes and lists. System designers need to prevent the need for typing and reduce error rates by using widgets.
- Sort, search and filter: Special considerations are needed for mobile apps in order to reduce the workload and typing input. In addition, the small screen size makes it easy for the user to lose context, so attributes like typo tolerance, predictive contextual help would be desirable.

Going forward to model these mobile apps usability from a product quality perspective, the table below shows a condensed version of 2Q2U with characteristics which can then be decomposed further depending on the domain and product goals.

Rapise Rapid & Flexible Test Automation - Click on ad to reach advertiser web site

Need to Test Your Application on Multiple Environments?

Writing Test Scripts Too Slow?



It's time to try a better way.

Rapise

RAPID & FLEXIBLE TEST AUTOMATION

Learn more at: inflectra.com/rapise

www.inflectra.com
sales@inflectra.com
+1 202-558-6885

inflectra

Characteristic/Attribute	2Q2U v2.0 Definition
1 Usability	Degree to which the product or system has attributes that enable it to be understood, learned, operated, error protected, attractive and accessible to the user, when used under specified conditions.
1.1 Understandability	Degree to which users can recognize whether a product or system is appropriate for their needs. Note: Same ISO 25010 definition.
1.1.1 Familiarity	Degree to which the user understand what the application, system's functions or tasks are about, and their functionality almost instantly, mainly from initial impressions
1.1.2 Control icon recognize-ability	Degree to which the representation of the control icon follows or adheres to an international standard or agreed convention.
1.2 Learnability	Degree to which the product or system enables users to learn its app.
1.2.1 Feedback Suitability	Degree to which mechanisms and information regarding the success, failure or awareness of actions is provided to users to help them interact with the application.
1.2.2 Helpfulness	Degree to which the software product provides help that is easy to find, comprehensive and effective when users need it.
1.3 Operability	Degree to which a product or system has attributes that make it easy to operate and control. Note: Same ISO 25010 definition
1.3.1 Data Entry Ease	Degree to which mechanisms are provided which make entering data as easy and as accurate as possible.
1.3.2 Visibility	Degree to which the application enables ease of operation through controls and text that can be seen and discerned by the user in order to take appropriate actions.
1.3.3 Consistency	Degree to which users can operate the task controls and actions in a consistent and coherent way.
1.4 User Error Protection	Degree to which a product or system protects and prevents users against making errors and provides support to error tolerance.
1.4.1 Error prevention	Degree to which mechanisms are provided to prevent mistakes.
1.4.2 Error recovery	Degree to which the application provides support for error recovery.
1.5 UI Aesthetics	Degree to which the UI enables pleasing and satisfying interaction for the user. Note: Same ISO 25010 definition.
1.5.1. Text color style uniformity	Degree to which text colors are used consistently throughout the UI with the same meaning and purpose.
1.5.1. Aesthetic harmony	Degree to which the UI shows and maintains an aesthetic harmony regarding the usage and combination of colors, texts, images, controls and layouts throughout the whole application.

Note the hierarchical tree structure of the model whereby one characteristic is defined (and evaluated and measured) by other sub-characteristics. Sub-characteristics can then be decomposed to attributes that can be directly measurable. Now, with a product quality model, we need to define a QinU model which will characterize the user's experience both from a usability point of view (efficiency and effectiveness at completing tasks) and a UX view that incorporates satisfaction and its many elements.

Using the Model in Real Life for Improvement

Up to now, we have discussed the special contexts of mobile software that require a model-based approach to evaluate mobile user experience and have set forth a model that can be flexibly used to suit our purpose; evaluation and improvement. Now, we demonstrate how to use the model and step through the process of doing a real evaluation on the road to improving a mobile application's usability and UX. First, reviewing the relationships in ISO 25010 in Figure 1, we can see that the external quality has an influence on the final effect of the software product, namely the Quality in Use (QinU).

In previous research, we used the ISO 25010 as a starting point, and developed 2Q2U (Quality, Quality in use, actual Usability and User experience) v2.0 [2 and 3], to instantiate the quality characteristics to evaluate and conduct a systematic evaluation using the 'depends' and 'influences' relationships. Now, taking relevant characteristics of the model, we examine them in terms of a requirement tree as shown in Figure 2.

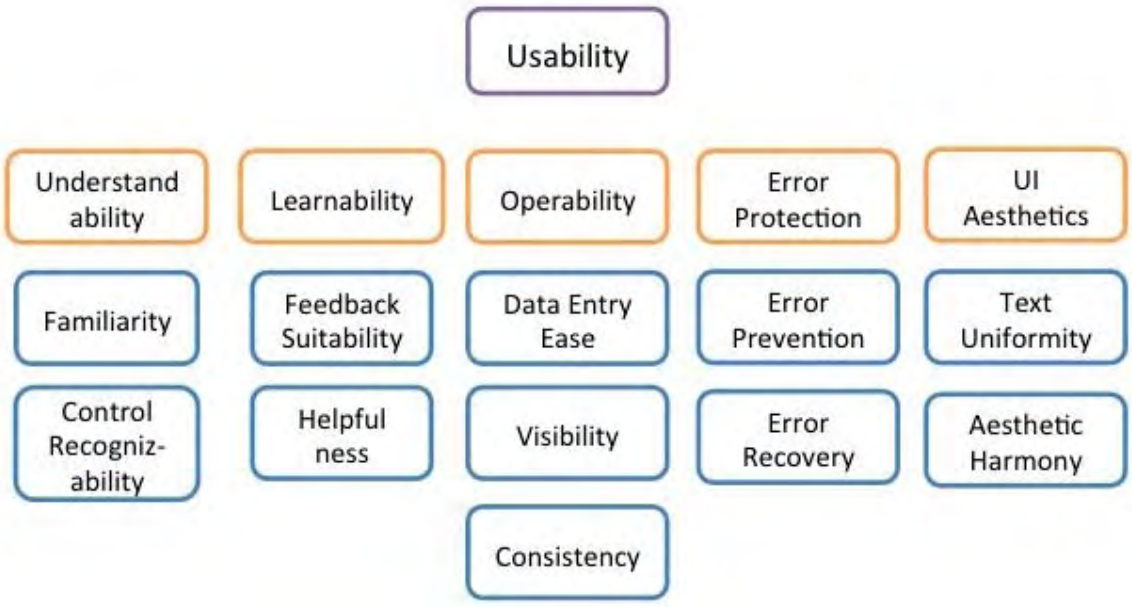


Figure 2. Hierarchical Product Usability Model (External Quality)

Note the hierarchical tree structure of the model whereby one characteristic (i.e. Operability) is defined (and evaluated and measured) by other sub-characteristics (Data Entry Ease, Visibility, Consistency). These particular sub-characteristics are then decomposed to attributes that can be directly measured. This represents a product quality model where we can measure a particular mobile application with inspection.

In this example, if we want to evaluate the Data Entry Ease for a mobile application, we need to specify a particular task and attributes that we want to measure and evaluate. Let's look at the Amazon mobile application (figure 3) as an example.

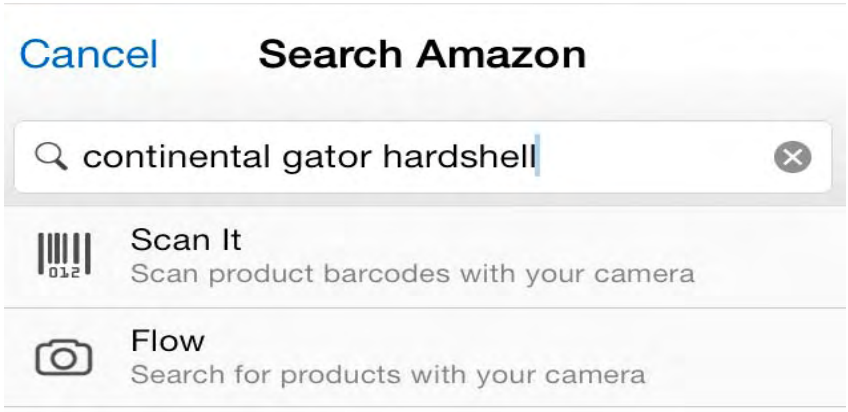


Figure 3. Amazon Mobile Application

Task: Search for Continental Gator Hardshell Urban Bicycle Tire

Attributes for Visibility could include:

Contrast: Ability to maintain contrast ratio of >=10 using a specific tool (i.e. http://www.snook.ca/technical/colour_contrast/colour.html)

Attributes for Consistency could include:

Font type consistency: All font types are the same on one page (3=complete, 2=partial – all but 2, 1=partial all but 2 or more, 0=mostly or all different font types)

We then evaluate these attributes given the particular task.

Examining the visibility attribute "contrast" we can see that the Amazon mobile site is black print on a white background. They certainly don't try to be fancy (so they may score lower on aesthetic attributes), but for being visible, they do well with a contrast ratio of over 20.

Examining the consistency attribute, font types consistency, we see they consistently use the same font, Arial, as well, so they score the maximum on this attribute.

Note that this evaluation is heavily dependent on the task at hand, and the attributes we choose to evaluate. If we chose aesthetics we may have other criteria where Amazon would not fare so well.

Once we measure these product quality characteristics and attributes, we can then determine its impact and influence on the end user's QinU, for instance, their efficiency in getting something done. Figure 4 shows QinU model for this example.

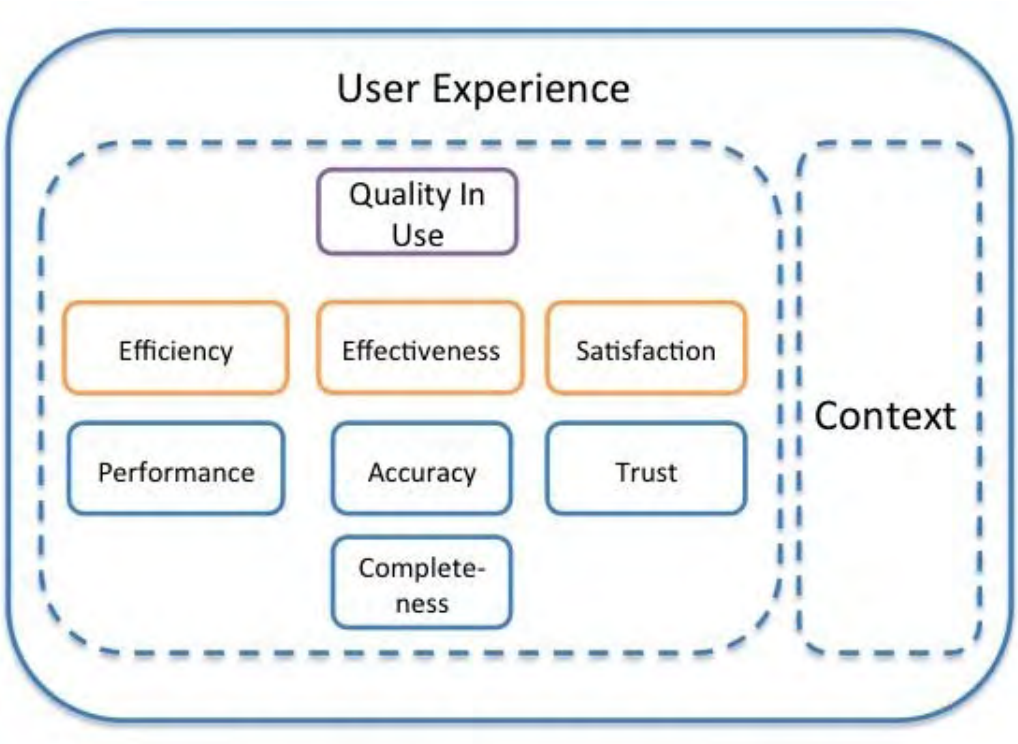


Figure 4. Hierarchical Quality in Use and UX Evaluation Model

In previous research, we have noted that User Experience includes 'softer' characteristics such as the satisfaction attribute [2,3]. Other researchers [5,6] have also noted the hedonic nature of UX in that it should at least be partially evaluated from feelings of trust, and attributes such as sense of community, rather than harder attributes such as performance (time to finish a task). So UX, while hard to define, can be defined roughly in terms of satisfaction and software attributes assigned to the satisfaction characteristic. As always, context plays a critical role in the evaluation of User Experience. This is one of the elements that sets UX for mobile applications apart. Contextual factors for mobile applications vary much more than for desktop applications. Contextual factors could include [4]:

- Activity: What the user is doing at the time of usage has a significant influence on the user's attention span. For example, if they are driving, then they have a very short attention span, maybe 1 second, versus if they are in the middle of a conversation, perhaps they have an attention span of 3 seconds. On the other hand, if they are at home sitting on the sofa, then the user may have other distractions as well.
- Day/time of day: The day and time can impact what a user is doing, and the level of natural light. Unlike desktop or Web apps that are typically accessed indoors, the usage of mobile apps is particularly sensitive to this contextual factor influencing visibility.
- Location: The location of the user influences many elements. For instance, indoors, outdoors, in a car, in an elevator, train or factory all of which can also be related to the user activity.
- User profile: The increasing complexity of software combined with an aging user demographic has an interesting affect on the usability of mobile apps. For aging users, usually their close range vision capability has diminished along with their dexterity. On the other hand, applications have become complex, and therefore function and content simple-ness and understandability is also critical and influenced by the particular user group. Not only are there more aging users, there are also younger users as children these days begin using computing devices as toddlers.

Going back to Figure 1, we can now modify it to match our current evaluation and get some insight and basis for conducting experiments and evaluations.

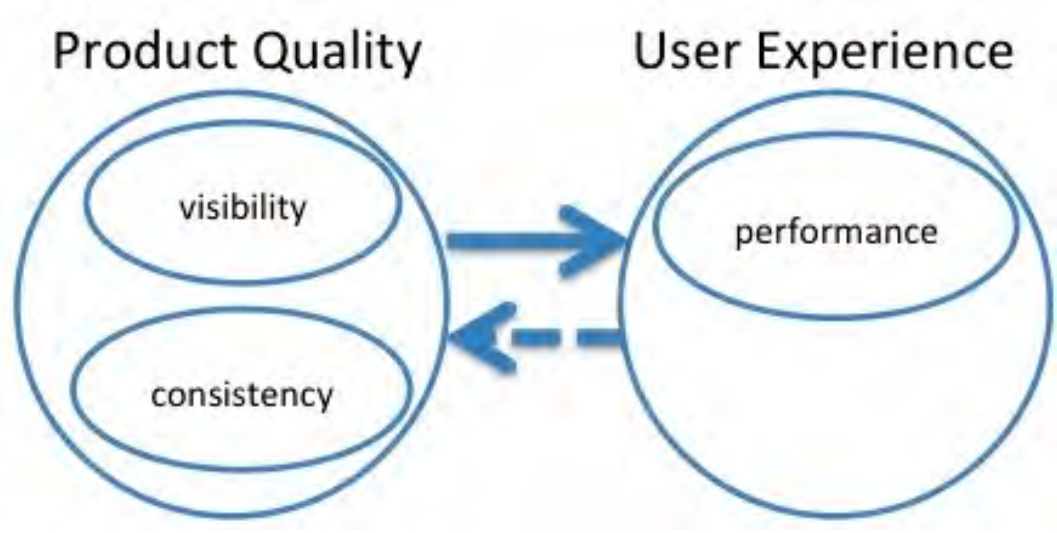


Figure 5. Product quality influences User Experience Dependency Model

We have measured visibility in our example above. Based on changes that we make in product quality attributes, what impacts will those changes make in User Experience. Suppose for example, that we chose a color scheme that changed the contrast to 7. Would that result in different user performance?

Using this causal relationship model, by changing the values of attributes in product quality and then measuring the effect of those changes in UX (possible through user logs, survey, or observation), organizations can iteratively understand what their users like or dislike and what makes them more productive or not, and then begin to make systematic improvements as a result of their understanding and knowledge gained. Of course, the trick for mobile is all the contextual factors as well.

Conclusion

In this article, we have discussed differences and peculiarities of the mobile context that warrant the need for a model-based approach to evaluate mobile user experience. Based on these requirements, we instantiated the ISO 25010 software quality model for the specific purpose of evaluating and improving the User Experience (UX) of a mobile software application. Using a structured and iterative approach, we have shown how it is possible to improve mobile application usability and user experience in a systematic way in order to somehow quantify as least parts of the ‘wow factor’ as a description of quality for the end user.

As time goes on, certainly, just as the ‘web’ and Internet applications matured with Web 2.0, the mobile platform and its applications will also mature. Standards, most likely, de facto, will arise that dictate how user interfaces will be built and thus partly leading to a standard UX associated with those interfaces. Until then, it will be exciting to see developers and designers pave the way as they create user interfaces that not only get the job done, but also create an experience that supports users in getting their jobs done while satisfying their hedonic needs. It is these designs that will succeed and separate ordinary companies and their mobile apps from the masses.

References

1. ISO/IEC 25010: Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models, (2011)
2. Olsina L., Lew P., Dieser A., and Rivera B.: Updating Quality Models for Evaluating New Generation Web Applications. In Journal of Web Engineering, Special issue: Quality in new generation Web applications, S. Abrahão, C. Cachero, C. Cappiello, M. Matera (Eds.), Rinton Press, USA, 11 (3), pp. 209-246. (2012)
3. Lew P., Olsina L., and Zhang L.: Quality, Quality in Use, Actual Usability and User Experience as Key Drivers for Web Application Evaluation. ICWE 2010: 218-232
4. Lew P., and Olsina L.; Relating User Experience with MobileApp Quality Evaluation and Design. LNCS 8295, Springer, Current Trends in Web Engineering, ICWE Int'l Workshops, Q.Z. Sheng and J. Kjeldskov (Eds.), Aalborg, Denmark, pp. 253–268, (2013)
5. Bevan N.: Extending Quality in Use to provide a Framework for Usability Measurement. LNCS 5619, Springer, HCI Int'l 2009, San Diego, USA, pp. 13-22, (2009)
6. Hassenzahl M.: User Experience: towards an experiential perspective on product quality. In: 20th Int'l Conference of the Assoc. Francophone d'IHM; Vol. 339, pp. 11-15, (2008)

Background Reading

- Lew P., Olsina L., Becker P., and Zhang, L.: An Integrated Strategy to Understand and Manage Quality in Use for Web Applications. Requirements Engineering Journal, Springer London, Vol.17, No. 4, pp. 299-330, (2012)
- http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=35733
- Bevan N.: Extending Quality in Use to provide a Framework for Usability Measurement. LNCS 5619, Springer, HCI Int'l 2009, San Diego, USA, pp. 13-22, (2009)
- Hassenzahl M.: User Experience: towards an experiential perspective on product quality. In: 20th Int'l Conference of the Assoc. Francophone d'IHM; Vol. 339, pp. 11-15, (2008)

Developer Careers Considered Harmful

Christian Grobmeier, cg @ grobmeier.de, http://www.grobmeier.de

When I published “The Zen Programmer“ book in 2013 [6], a manager from a big company sent me an email. In this book I describe how I applied Zen to my daily work and how I am able to keep balance despite how hard I work.

The manager told me that almost everything I wrote was just plain wrong. For example, I believe that there is nothing like a career one can have; he responded that my career would be successful if I just would do what my manager commanded. “Jump if your boss tells you to jump”, he wrote.

This certainly is not how I want to live. A career is an illusion, a theoretical term, but Life is not. “Making a career” is sometimes just the wrong thing to run after, and that can cause serious harm to you.

At least in the western world things are running just fine for software developers. We have a choice and we have some freedom. The tools we use and the expectations towards developers are not longer as they were twenty years ago. But the way to work didn't change as much.

Today we are told that we need careers, constant happiness and are never allowed to fail because otherwise terrible things would happen. But the statistics on depression and burn-out prove us there is something wrong with this thinking. In the following text, I will elaborate on a few of my thoughts and explain, how getting rid of my career made it more “successful” than ever.

The way we work

In 1980 work and life was different than it is today. This can easily be proven if you just look at the tools that we use today: tablets, smartphones, mobile computers. You can work everywhere, even on the toilet, and people do work everywhere. Life and work today can't be separated that easily anymore. I wrote parts of this text when waiting in the car to pick up my son from the kindergarten. Twenty years ago I most likely would have just read a fiction novel.

For that reason we cannot say: we worked hard in the 80's already, why do people start to complain about that now? The reason might be tied to the fact that emails didn't exist back then and nobody expected to receive a response late at night or on weekends.

As a society, we have reached a new milestone: work and life become one single thing. We cannot ignore a change and just keep on going. That's what dinosaurs probably did.

We need to deal with modern working on the one hand, but we need to do it right. Otherwise we will have a hard-to-beat enemy: stress.

Stress may have a negative impact on our productivity, life experience and can even cause physical problems like heart attacks or strokes. It can lead even further to mental diseases like depression or the burn-out syndrome.

Agile Development & Better Software Conference East - Click on ad to reach advertiser web site

Some people claim depression can be cured with just pulling oneself up and burn-out doesn't exist. But on the other hand, well-educated medical personnel are treating people with these problems. And that is a good thing: mental illness exists, and diagnosis or getting rid of it is difficult. Ignoring their existence may lead in a downward spiral, which ultimately can even lead to suicide.

Statistics say, taking sick leave from work is on the rise due to stress. Mental disease costs companies a lot of money each year. In Switzerland, the cost of mental illness is around 15 billions of euros each year [1]. In Germany it's similar, and there is one more concerning fact: according to the TK (a German health insurer) mental illness is not only a disease which takes the most time to cure but has overtaken any other reason to stay away from work.

As employer, but also as employee, we can't ignore that. Workers need to stay healthy if we want to avoid risks for the company. What can happen is not only losing employees with high expertise but also losing reputation worldwide. The bad news on working practices at Foxconn did not only have an impact on the company itself but also on their main customer Apple [4]. Apple executives were forced to react, which alone is awkward.

My father told me I need a career

Some people may feel pity for me, as I would “never make a career” in this life.

Why should I make a career? Some people seem to think you can life a “happy life” only with having “a successful career”. Looking closer at “happiness” and “career” I couldn't find that much substance behind these terms.

Some say you'll be very happy when you have “made it”. It's a kind of binary decision, you make it or not. It is often not further defined how you can “make it”. However it seems to be clear that drug dealers, burglars or solitary alcoholics haven't made it. But a dentist driving a Porsche, a software manager with a Rolex or a politician working in the US Congress are considered the opposite. It's getting difficult if we think about a (stressed but happy) mother of six or a farmer, who likes his job. Maybe these people are happy, but usually nobody would say: the farmer really made it! He has a job he likes! Or, look at the tired mom around the corner - she must have made it!

If one could choose a life right now, I think most people would take the Rolex, the Porsche or the job with reputation. I showed this list to a few people and many would have picked the Porsche. I did not say the dentist was happy. I asked for the reasoning and it seems people expect Porsche-drivers to be happy, and being a dentist simply sounds better than being a mom.

With that observation in mind, it looks like we need a successful career to “make it”, and we need to make it to live an easy, happy life.

It would be OK for some people, to first work very hard (like studying dentistry) and after a while they can enjoy the rest of their life. That's exactly what my father told me: if I didn't learn much and work hard I would end up as a poor assistant to some random craftsman. This vision combined the worst two scenarios for him: being poor and having a job without reputation. He told me that I would never be happy without earning a lot of money.

Agile 2014 Conference - Click on ad to reach advertiser web site

“What is a happy life?”, I asked my father.

He couldn't give me a clear answer, but it included the ability to travel to exotic places, owning a home, and to be able to have dinner in a restaurant from time to time. If that would be the only thing we need and want from life, then it is a logical step to run after a career.

But there are fine nuances. My father would have told me, that owning a Porsche is too expensive and only a few people will be able to buy it. And he is at least not wrong with that. There are many people out there who will never “make it”. And I speak of well-educated, nice people, who fail at some point in their career.

It would eventually mean that only a few of us are able to live a happy life.

The career model and its flaws

The career model is like a pyramid, and you need to get up the ladder to earn more money. As long as you are on the bottom of it, you have nothing to laugh at. But so you move up more, and the richer and happier you'll become. It's a carrot system: first do the work, then take the money.

I would guess it needs around 20 years of hard work first to succeed in this model. As a dentist you need to study, then work and pay back the loan. If you want to open your own clinic you have to buy a lot of expensive devices. You also take quite a risk with employing dental nurses. I calculated that it would take around €500.000 to make a person a dentist, if not more. In Germany it means that you are around 50 until you are out of the biggest debt. Then you have “only” 15 years left to “make it” before you retire.

But there are a lot of risks: a drunken driver could hit you when you leave your favorite coffee store. Is that unlikely? It happened to a colleague of mine recently (although the driver was texting).

Honestly, a lot of things happened recently that were considered almost impossible:

- in 2011 around 20,000 Japanese residents were killed by an earthquake and a following tsunami, which caused nuclear meltdowns in three nuclear reactors. From the new dead zone, 150,000 people were expelled from their homes. Almost uncovered by media were the volcanic eruptions in the same time happening in the south of Japan. Until today the situation is not under control with tons of radioactive water spilling out from the reactors day for day.
- In 2014 – after Russia accepted Crimea as part of their federation – people started to speak of a Cold War again. The country is deeply divided and if it hasn't happened yet; there is a new war taking place in Europe.

Life doesn't run straight and unexpected things can happen. Nobody expected natural disasters of that strength and nobody expected a new war in Europe. Only a few people expect their own death.

What I want to say is: when something is so uncertain, so fragile as life, how can we make such deals as “working hard for 20-30 years” just to have a couple of “happy years” after? The career model isn't accepting that we are humans. It's like Russian roulette: hope you do not die before you have managed to get a certain amount of money.

Being unhappy is OK

We all do this because we are constantly told that being “unhappy” is something which we need to avoid. We waste a lot of our lives to reach permanent luck and wealth, but in the end we'll find out that even the richest of us do not constantly smile. Here is a secret: even George Clooney sometimes poops. And sometimes his teeth hurt.

No matter what today’s media tells, it's perfectly OK to be unhappy. Nobody feels happy all the time. Corporations play with our fears to make us buy their products. But it's not that their products would help us to prevent sickness or solitude. If it would just be that easy.

At the university I learned people are constantly trying to imagine how others look at them. This may develop self esteem (or not). What, if we all just buy these things to make others think we are “happy”? This would certainly be very stupid, but we all are struggling with this. If we would have a great job, then people would surely think I am a healthy, intelligent, high-performer. If we have the job of a cleaning lady, people may think I was too stupid for college.

We are afraid to be alone, and this is what social media and our cell phones fix.

When I travel for work I often take the tube. Recently I was the only person in that wagon who didn't look at the small screen of a smartphone so I could observe others. None of them looked happy in that moment. People are constantly distracted. If we are not participating in social media, we might feel to miss something. We don't want to be disconnected from the people around us, so we stick with sending unimportant messages to people we rarely know. Just to not feel alone.

Were people less solitary without their smartphones? I doubt it.

In fact we are slowly unlearning to accept that life is not only happiness, it's also being unhappy. Unhappy people like Edgar Allen Poe wrote about in some of the best poems ever written. It's not necessary to avoid this, it is to accept.

Sometimes humans are feeling solitude. It's natural and we don't need to fix this immediately.

It's also OK to feel bored. As Nietzsche once said: “Boredom is the lull of the soul,” (freely translated).

We need to accept that these feelings are there. We should not overvalue them. We should also not stick with these feelings. We need to learn: all happiness will go at one time. But also unhappiness will go. Both are tied to each other. There is not only one thing to achieve, it's always both.

The career model doesn't accept we are having these two sides in us.

Hippies? Weaklings? Wise?

A group of people, often called “Y Generation”, are people born between 1980s and early 2000s. They want to keep a balance between work life and free time. They want to develop themselves freely, work fewer hours than usual and maybe think, “working” was done wrong in the past. When they got attention recently in German press, they were sometimes called weaklings because people assumed they are lazy or too soft.

It seems that there is a demand of some people to change the current way to live. Actually I agree, when they say we don't have a healthy relationship to our work today. However, having a lot of spare time would be just another way to run after a career. There is a time for fun, but there is also a time for work.

“A day without work is a day without food,” Kôdô Sawaki

As it is wrong to say: “I want to work 60 hours each week”, it is also wrong to say “I want to work no more than 10 hours each week”. Sometimes it just feels good to work hard. Sometimes it's bad. With dogmatic views on work times, we miss one important thing: work is part of our lives.

It's not only exhausting and causing stress, it is sometimes also fulfilling and satisfying. I met one of my best friends at work. We are friends, since we made a big project work just fine, when everybody thought it was a lost cause. It was exhausting, but also fun to work there.

The “Y Generation” wants to experience life. But sometimes life requires long working days and even stress. The right view would be to understand that these periods will go away. And if there is no change in sight, or when one cannot identify any longer with what he does, there is most always a way to leave the situation. I believe it is not the hours we work which cause the problems; it is how we look at our work and if we believe it is worth doing it.

Somebody once asked me, if Zen wouldn't mean to just quit the job and become a monk. For some of us, it is the right thing to do. For some not. Zen does mean to do the right thing, and it's up to you to decide what the right thing is. Personally I am not ready yet to become a monk. I struggle with a lot of things, and I admit, I am fighting the fear of tomorrow as many of us.

The discussion on self-expression and and how many hours you work or not is again not fixing the problem, that at some times you feel energetic and at some times not.

“Every day is a good day”, I learned from Kôdô Sawaki. To achieve this, we often do not need to change the method, but we need to change the view. When we have changed the view, the method will adjust. Extreme views in any kind are rarely helping.

I am afraid

We worked hard before, and with the arrival of tablets we work even harder. Being online all the time, it became harder to speak to the person standing next to us. It's possible to fix a business deal when our counterpart sits on the toilet. In the tube we can chat with mom and the last thing we may do before going to sleep is checking our business emails.

Why?

Maybe because we are afraid.

You could lose your job.

You could lose your life partner.

You could lose your kid.

You could become poor and solitary.

You could not get a promotion.

You could die.

In fact, you will die. As everything else on this planet. Some people tend to think dying is a bad thing which we need to prevent sometime in future. I don't think so. I am afraid to die myself, I hope it will not be too painful. But in the end, it could be more worse to stay alive than to die.

In ancient times, men were afraid before the darkness. Today we have put a candle in each corner of the earth which gives us light. With social media, making careers, considering ourselves important, we do the same: putting up candles at places where we are afraid.

Slowly we melt more and more with the technologies we build, becoming some kind of cyborgs – see Google Glass [6]. Or more drastically, we have started to distract ourselves at any time and anywhere, with business emails, social media, taking pictures. We have started to lock ourselves into a digital world which in the end makes us live like zombies.

We need to get out of this circle of fright.

You don't need to earn happiness

The people who told you that you'll be happy after working hard for 30 years were wrong. The people who told you that you need to be successful at your job were wrong as well. The people who told you that you need to self-express at any price were wrong. The only truth is: you need to breath – and maybe a couple of other biological things.

Whoever you are, where ever you live, how much you ever have worked: you deserve to be a happy person. And there is absolutely nothing which prevents you to be one. Except that you will struggle hard with this thinking when you have to fight for existential things like water, food, or safety.

I am honest: I want to own a house. Maybe nearby a sea with a great view, or close to the ocean. I know, at least in Europe the cost for that house would be insane. I would have to work for the rest of my life to pay back the loan. For that reason, I have decided this is not the way which would make me happy. I don't want to commit myself for living a long time at one place. For you this may be different. For you it might be the right decision and you will happily take on the burden. But if not, then there is absolutely no need to get such a house just because you were told to do so. It will not make you happy afterward, it needs to make you happy right now, when you decide to spend your life for it.

Although living a simple life in a rented flat and working hard every day including Saturday and Sundays I am happy. Because I do what is right for me.

How can you work so much?

To be honest, my workload looks insane to others sometimes. I try a lot of things, and I have a family. It's complicated to get everything under one hood, but it's doable. At busy times, my day starts at 5am and ends at midnight.

Zen changed my view, but it's a hard practice. You can't learn Zen from a book, it's something you need to do every day. It's difficult to do a sitting meditation in an open space office, but inspired from Kôdô Sawakis words, that one can do Zen at anytime, I created my own strategies to practice. I call it “Kizen”. It's best translation maybe would be: “The way of Code”. I would not go as far to replace other meditation practice with Kizen. But I feel it's a good companion because you can do it at any time.

One of my favorite Kizen practices is to drink tea. Actually I love coffee very much and I drink lots of it in the morning while working. As this is nothing extraordinary to me, I have chosen to do something else which will need my full attention. Almost once a day I try to brew and drink the best tea ever made. I take time for this event: I clean the place before I make tea and to clean up everything afterward. Perfect tea needs more than perfect leaves. It needs the right temperature of water, the exact time to brew. It's even more complicated to drink tea: one needs to concentrate on the aroma, the taste and the perfect size of the nip.

With fully focusing on the tea, I can get rid of constantly thinking on today's business. My mind gets a distance and I calm down. Doing this and some other practices as described more in detail in "The Zen Programmer" help me to get things done without being worried about my workload.

A career, without a career

Previously I wrote that running after a career is bad, but then I explained how I can manage to get done with an unbelievable huge amount of work. As mentioned, it is the point of view you take. Anybody who read this article that far could argue, that I would actually use Zen practice to increase my working performance to ultimately make a career.

In reality I consider "successful career" a side effect which you may have, or maybe not. Since I learned that my career doesn't need to go into a specific direction, it's running on steroids. I am having some kind of career, without wanting one or caring about it.

Let your career go. Instead focus on what you think is the right thing to do.

Life is not in ten years. It's now. Every second.

When I understood that my life and my happiness didn't depend on my career even in the slightest way, I started to lose my fright before the future. Nobody knows what the future brings. It doesn't make sense to be afraid before it. Without fear, I was able to make the decisions that changed my life in the way I have it today. Maybe still full of work, but with work of which I know that it is the right thing for me. With that knowledge I can enjoy the good and the bad parts.

Ask yourself the questions:

- Do you like what you do?
- Do you think the things you do are things you want to do?
- Would you do the same things if you were to live alone, without kids, no loans or other things you might be responsible for?

If the answer is "yes", then stop running after your career. You have already reached everything to be happy. Just continue to do the best you can, but be aware that things are good as they are right now.

If your job is not like that, then you might start to hate it one day. You'll be worried about your situation. You might panic when somebody else is able to "make it", but not you.

> "No one is born hating another person because of the color of his skin, or his background, or his religion. People must learn to hate, and if they can learn to hate, they can be taught to love, for love comes more naturally to the human heart than its opposite." - Nelson Mandela, Long Walk to Freedom

When I started with meditation I lost the ability to hate. I relearned to breathe. Eventually I could see that it is only me who am responsible for my life.

- What we should understand:
- A project is just a small slice of your life. There is your family, nature, the air you breathe.
- A missed a deadline, requires action, not worries.
- Your happiness doesn't depend on your job. If you lose it, it maybe be uncomfortable but it is unlikely that you die alone, poor and suffering.
- Your boss might lead the project, but not your life.
- There is no reason to be afraid for the future. All future is uncertain.

Leaders, and Leaders

I know quite a couple of people who were working, and almost every year they would earn a promotion. At lower levels the promotion is almost automatic. You can reach a career level by just not quitting your job, as long as you are half-competent in what you do. This alone is quite absurd.

Some programmers think their career goal is to quit programming and become a manager too. The difference between these two jobs is that big, one cannot say its "promoting to an upper level". It's switching jobs.

In 1969 Dr. Laurence Peter wrote something which went famous as the "Peter Principle":

> "In a hierarchy every employee tends to rise to their level of incompetence," - Dr. Laurence Peter

It means that you become promoted, until you are no longer able to deliver satisfying results. At this point promotions would stop and you would stick with a job in which you can't perform well.

There is quite a chance that you are very good at programming, but not that good with managing a company. Working with people is simply not the same as writing computer code. Still people aim to reach higher levels and are even willing to sacrifice their happiness.

There are good team leaders by heart. Like Sun Tsu described one in his excellent book "Art Of War" [3]:

> „During the Warring States era, when general Wu Qi was military governor of West river, he wore the same clothes and ate the same food as the lowest of his soldiers. He did not use a mat to sit on, and he did not ride when traveling. He personally carried his own bundle of provisions and shared the toil and hardships of the soldiers. Once, when one of the soldiers was suffering from a festering wound on his arm, the general himself sucked out the pus. When that soldier's mother heard about this, she began to mourn. Someone said to her, "Your son is a soldier, yet the general himself sucked the pus from

> his wound - what is there to mourn about?" The woman said, "Last year General Wu did the same thing for my husband, and as a result my husband fought in battle without taking a backward step, finally dying at the hands of an enemy. Now that the general has treated my son in this way too, I have no idea where he will die. This is why I mourn him." - Sun Tsu: The Art Of War [3]

This kind of leadership can't be learned easily. Wu Qi didn't care much about his career level. He knew that a career is worth nothing. It is the goal that we need to achieve. The way he led people is nothing that can be learned easily. Such leaders would not be recognized in the current vision of career. Nor would a person who was semi-automatically promoted to the career level of a general be able to act like Wu Qi.

Decide yourself

Whether you decide to stick with your company or to leave it: what counts is to do the right thing. Happiness is not a constant thing. After periods of excitement and fun, you might be unhappy in each and every company. Things come and go, chances too. There is no reason to be afraid before the future.

You live now. You need to decide what you do with your life.

I have decided to not wait for happiness until I am promoted to a certain level in the career game. I don't worry on a successful career, I worry about a successful life. And that is not tied to working hours, it is tied to the ability to sleep when I am tired and to eat when I am hungry.

The career model makes people sick and a lot of us do fail at the game.

We need to forget that becoming a manager is the next milestone to reach for a programmer. Instead we should focus on the things we are good at and which we want to do. If you can feel the fun in it, then you are at the right job – and you should decline any further promotion which would move you “up to the next level”. Because there are no levels.

References

1. Cost of mental illness (in german): Last retrieved on 08.04.2014. http://www.tagesanzeiger.ch/leben/Psychische-Krankheiten-kosten-die-Schweiz-19-Milliarden-pro-Jahr/story/18005771
2. TK Gesundheitsreport, last received on 08.04.2014: http://www.tk.de/centaurus/servlet/contentblob/516416/Datei/83065/Gesundheitsreport-2013.pdf
3. The Art Of War: Complete Text and Commentaries, Shambhala Publications Inc, translated by Thomas Clearly.
4. China Contractor Again Faces Labor Issue on iPhones, last received on 12.05.2014: http://www.nytimes.com/2012/09/11/technology/foxconn-said-to-use-forced-student-labor-to-make-iphones.html?_r=0
5. Google Glass: http://www.google.com/glass/start/
6. "The Zen Programmer": http://www.zenprogrammer.org, or at Leanpub on https://leanpub.com/thezenprogrammer

TargetProcess - Visual Project Management

Franco Martinig, Martinig & Associates, http://www.martinig.ch/

TargetProcess is a commercial Agile project management tool that allows following a Scrum, Kanban or customized approach. It provides an intuitive and rich visual interface to manage your software development projects in a collaborative way.

Web Site: http://www.targetprocess.com/
Version Tested: TargetProcess version 3.2.2, tested during a period from May to June 2014
System Requirements: if you want to have TargetProcess insalled on-site, you can check requirements on http://help.targetprocess.com/troubleshooting/onsite_requirements
License & Pricing:

- Hosted: free until 5 users, 25 dollars/user/month above
- On site: $249 user

see http://www.targetprocess.com/pricing/ for details
Support: e-mail, live chat, help desk

Documentation

TargetProcess provides plenty of online help that is accessible directly from the tool. The help system presents articles on main topics and videos on major features. It has an efficient search function. When you start using TargetProcess, there is already a set of example data that allows you to understand how the software works. If you have additional questions, TargetProcess provides help desk support that was very reactive during the time of this evaluation.

Configuration

The configuration screens allow to define processes where you can start from scratch or clone an existing (Scrum, Kanban) approach. You can also define the project teams and members, the roles in your projects, business values like the priority of user stories, e-mail notifications. TargetProcess has an open architecture and existing plugins allow to cooperate with external applications like Bugzilla, Selenium, Visual Studio or Jenkins.

Concepts

The concept behind TargetProcess is rather simple: it takes the three main components of a project - the people, the work and the plan - and makes it easy to manage and associate them in a visual way. The work part is based on a classical agile requirements hierarchy that is composed of features, user stories and tasks. Two additional items are provided: the bugs and the impediments. The planning vision is also classical where the project can be decomposed in releases and iterations. On the people side, you manage user with roles and teams. You can have multiple teams working on the same project and they will share the complete view of the project.

The domain model is easy to visualize and understand. The strength of the product comes in its visual management capabilities where you are able to plan and track usually your project with a lot of different perspectives in a flexible way.

Approaches

This flexibility allows TargetProcess to provide initially with two different approaches that have their own processes: Scrum and Kanban. For both cases, you will have a set of predefined visual boards that are specific to each approach. You can define a default process for each new project.

Additionally, you can create your own customized process by cloning an existing one. A process is composed of practices (requirements, test management, bug tracking, ...) that can be associated to a process. You can add and remove practices as needed. You can define workflows for specific items in your process. These workflows can take the form of either a visual state transformation flow or a simple e-mail notification when for instance an impediment is created.

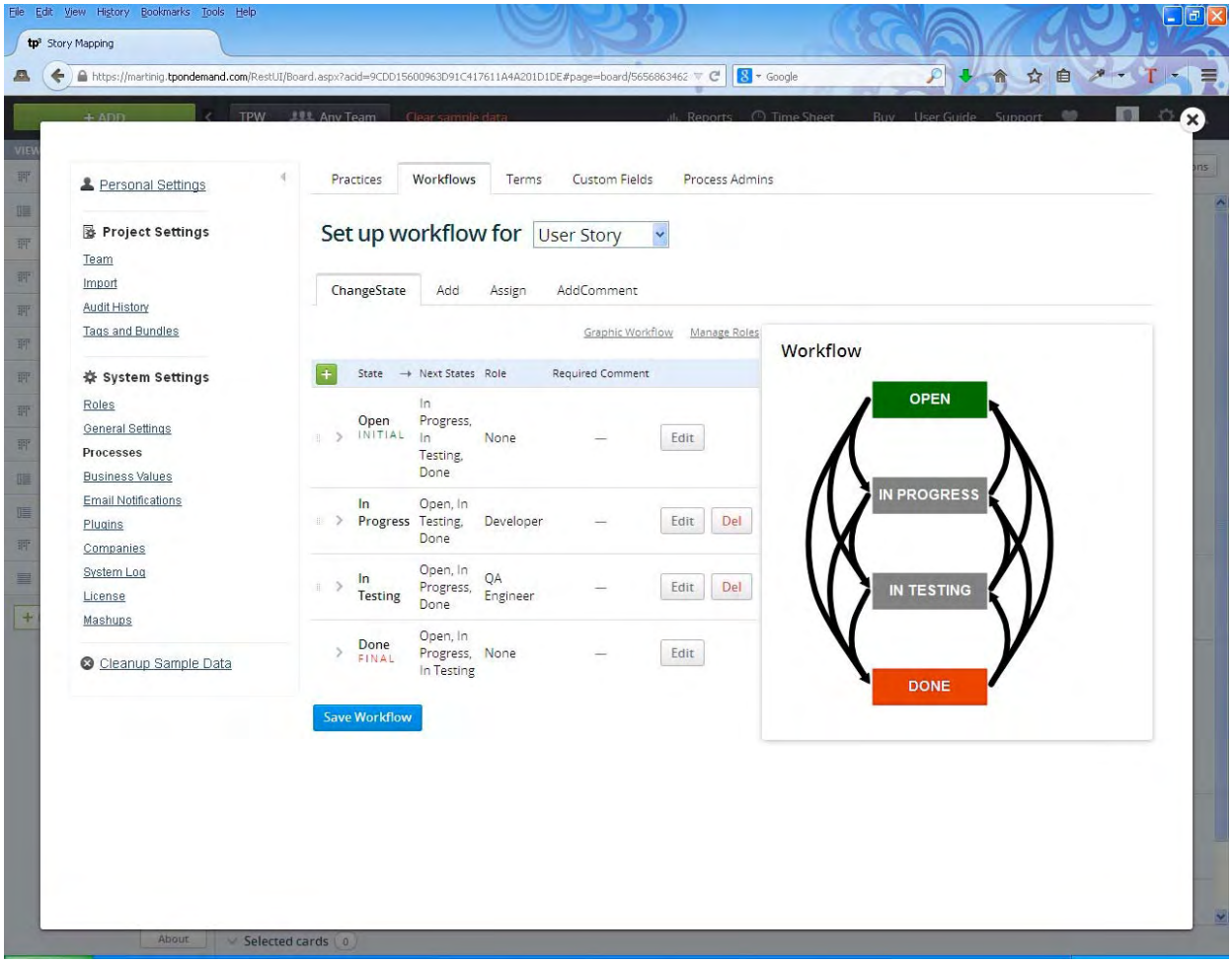


Figure 1.Definition of the workflow for the User Story item

The central screen of TargetProcess provides an easy way to interact and a quick access to every functionality. It provides also an easy way to monitor the status of your project and spot problems.

The screen you need for your task is rarely further than two clicks away from your current location. A large "Add" button at the top of the screen allowing to add quickly any item that you need to manage your project. Most screens use a tab approach that allows to manage different aspects easily. The visual boards, called "views" use a simple drag and drop approach.

If the views are visible as "boards" by default, you can also choose to display the cards as a list or as a timeline. This timeline view mode is one of the interesting features provided by TargetProcess, as it is very useful for portfolio management, roadmapping, release and iteration planning. You can also select multiple cards to perform actions on them. Besides the views available for each process, you can create your own views by cloning existing one or working from templates. The cards used on the views can be customized to display more or less information, depending on their type: feature, user story, task or bug. A tagging system is provided to add information to the cards. TargetProcess allows you also to define and limit the access to the different views.

If backlog items are created independently, in the real life there might be dependencies between features, user stories, tasks or bugs. You can manage these relations that are defined as dependency, relation or blocker.

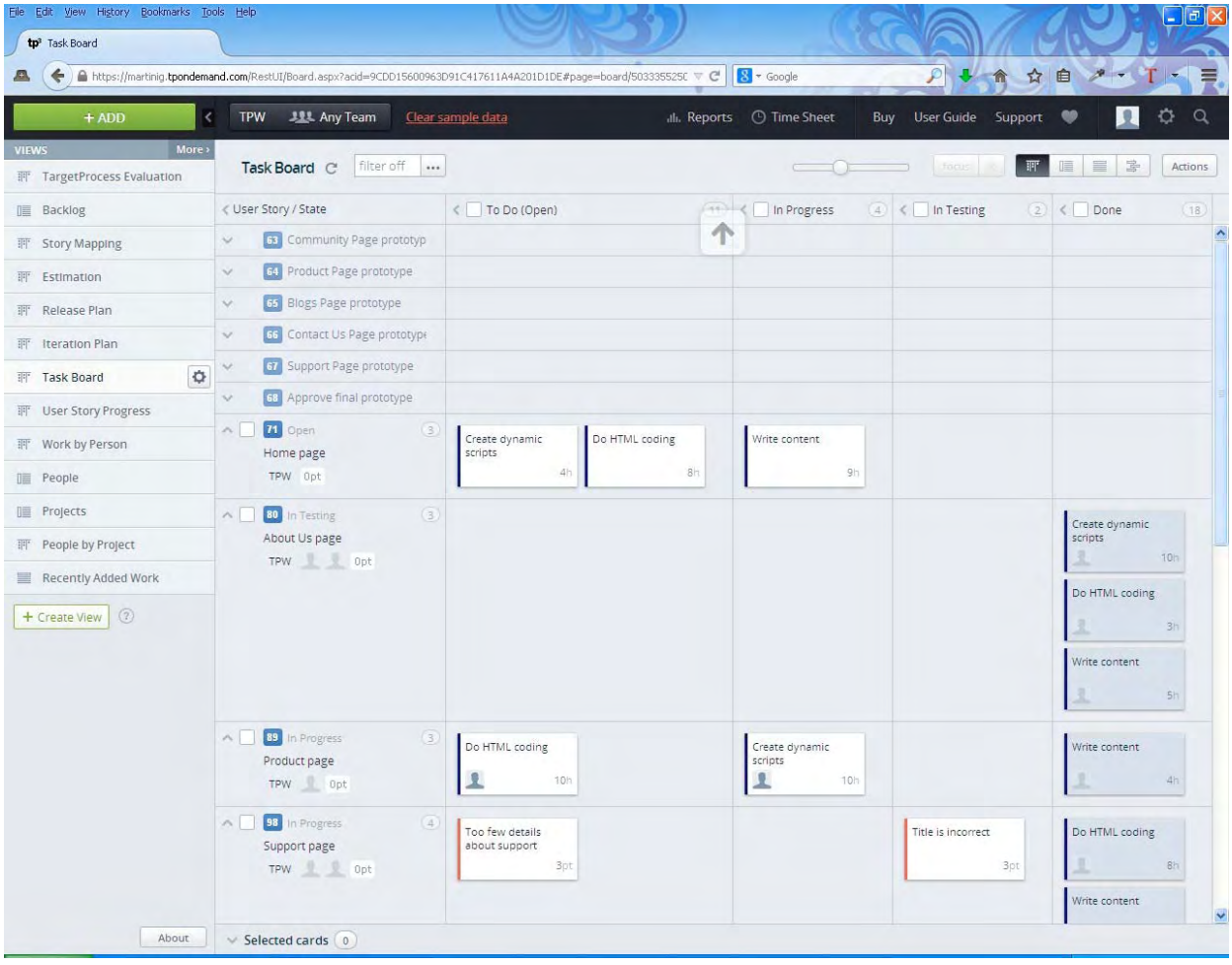


Figure 2. Iteration tracking in Scrum

In addition of its management boards, TargetProcess offer also a list of Agile reports that are useful for visually tracking the status of your project like the burndown chart or the process control chart that shows the distribution of cycle time of all completed entities (user stories, features, bugs, tasks, requests) for a given time frame.

Additional Features

Besides its core Agile project management features, TargetProcess offers a number of additional features that connect it with other tools for software development activities like source control, bug tracking or time management.

- **Import / Export**

TargetProcess features a two-way integration with Excel so that you can import and export data using the csv format. You can therefore load or update data about your features, user stories, tasks or bugs into the tool from an external source. You can also export data about these items to produce specific reports with Excel.

- **Integration with external software**

As far as configuration management is concerned, you can integrate TargetProcess with source control software like Git, Subversion, Microsoft's Team Foundation Server (TFS) or Mercurial. This integration is managed by plugins and allows sharing information about code updates.

On the software testing side, you can import test run results from NUnit, Selenium and the Jenkins/Hudson continuous integration tools. You can link external test results with an existing test plan and create a new test plan run. This gives you the capability to track in real time your daily build tests in TargetProcess.

Finally, TargetProcess offers integration with two IDE: Visual Studio and Eclipse/Mylyn that allows to share and update todo lists.

- **Time tracking**

The timesheet allow you to consolidate the daily working hours of the team members. It can be managed on one screen with all the tasks linked to a team member or it will be automatically filled when the time spent on a specific task is updated in the task management screen.

- **Bug tracking**

The bug tracking practice is completely integrated in the project management features of the tool. You can quickly add a bug item, which has its specific workflow, and manage it as a user story and task on your planning and tracking boards.

Conclusion

TargetProcess is both a simple and powerful agile project management tool. It is simple because it has an intuitive user interface and provides by default a complete Scrum and Kanban process for software development organizations that are transitioning to Agile. It is powerful as it allows to create a flexible approach to visual project management with the customization of item content, status and workflows. You will be also able to create your own views of your project status. The ‘visual’ aspects of the software and its customization features help the project managers to stay focused on what is specifically important for them and quickly spot problems.

It is easy to check if TargetProcess meets your needs, as a fully functional account up to 5 users is available for free without any time limit.

KADOS - Open Source Scrum

Charles Santucci, Marmotte Technologies, http://www.marmotte-technologies.org

KADOS is a web tool for managing Agile projects (Scrum more specifically) through visual boards on which are displayed post-its representing User Stories, Tasks, Activities, Issues, Actions, Bugs and any objects you wanted your project to manage.

Web Site: http://www.kados.info
Version tested: v1.7 released on May 1st 2014
System requirements: a web server with PHP 5.3 at least, a MySQL database v5.1 at least
License & Pricing: KADOS are published under MIT and GPL licenses and it's totally free
Support: by forum on Sourceforge.net or e-mail to the team.
Languages: French, English, Spanish, German and Brazilian Portuguese.
Documentation: Documentation is available online in French and English at the KADOS documentation web site

Introduction

Excel backlogs are nightmares

When they start to work with Scrum, most teams use Excel sheets to manage their backlogs. They are easy to build, to send by email and to update. The ScrumMaster, previously a project manager, is happy to keep his management tool and forgets that his job has changed with Scrum. The customer is also happy, as the IS/IT teams will manage the project. There is no need to be involved and all errors will be blamed on the lousy communication and the team!

But Excel sheets have a big default: they lack visual management ability. Visual management is currently done by sticking post-it on walls or big boards with any columns that fit your project.

Visual management is efficient … if you have walls

Then you have your wall covered with post-its. It is nice because there is a lot of colors that help you to represent functions or type of requests. Everybody shares the same view of the project and the team members move their post-it smoothly. But, one day the Scrum Master begins to stop updating the burndown daily chart. And team members are not so eager to move a task when it is finished. Then a management decision starts to slice your team between here and another country.

Where could you find a wall between here and there? It seems that the adequate wall is somewhere in the Red Sea. It is not easy to continue with visual management. Then somebody has an idea. Let's find a free, self-hosted, Scrum compliant tool allowing to keep our visual management efficient.

Why making a new Agile project management tool?

It not easy to find a free, self-hosted, with some support and regular releases for backlogs. There are many of them created since 2000. The problem is that you have to pay a fee for most of them or they are not free software or when they are, you don't have all features in the Community Edition ..and the missing features are just what you needed.

After a few hours peeling the web, when you find one that is promising, you just forget that it is written in Java and that you only have a PHP hosting available. At that point, you are faced with a sad fact: there is tool matching your needs! No problem, let's make it

KADOS distinctive feature

This introduction explains the main distinctive KADOS features: post-its on boards! To be efficient, post-its are managed on walls and this is inserted in a project organization consistent with Agile and Scrum approaches. Then, taking advantage of putting post-it in a database, why not use the boards for all post-its attributes. Most tools use columns board just to change the post-it status, but it's not that difficult to use board for setting priority, business value, complexity, hierarchy, etc...

This is why you will find a lot of “dashboards” in KADOS. Each one will perform an attribute change for post-its. Therefore, you have no lists or elaborate forms for managing post-its, just boards with columns! (*Excel lovers may have a small heart attack at that point, let's allow them to stop reading in order to get better*)

Installation

Go to SourceForge to download the last release. You just need a web server (Apache is the best) and a MySQL database with admin access to create the tables and insert basic data. Then you'll need at least one user with read/write rights for the application to access the database.

Follow the install instructions: deploy the application, launch the ".sql" file and set the connect.conf file to give access to the data for the KADOS application. Connect to the application using login *admin* and password *admin* (change the password if you are willing to use the application in production)

Projects

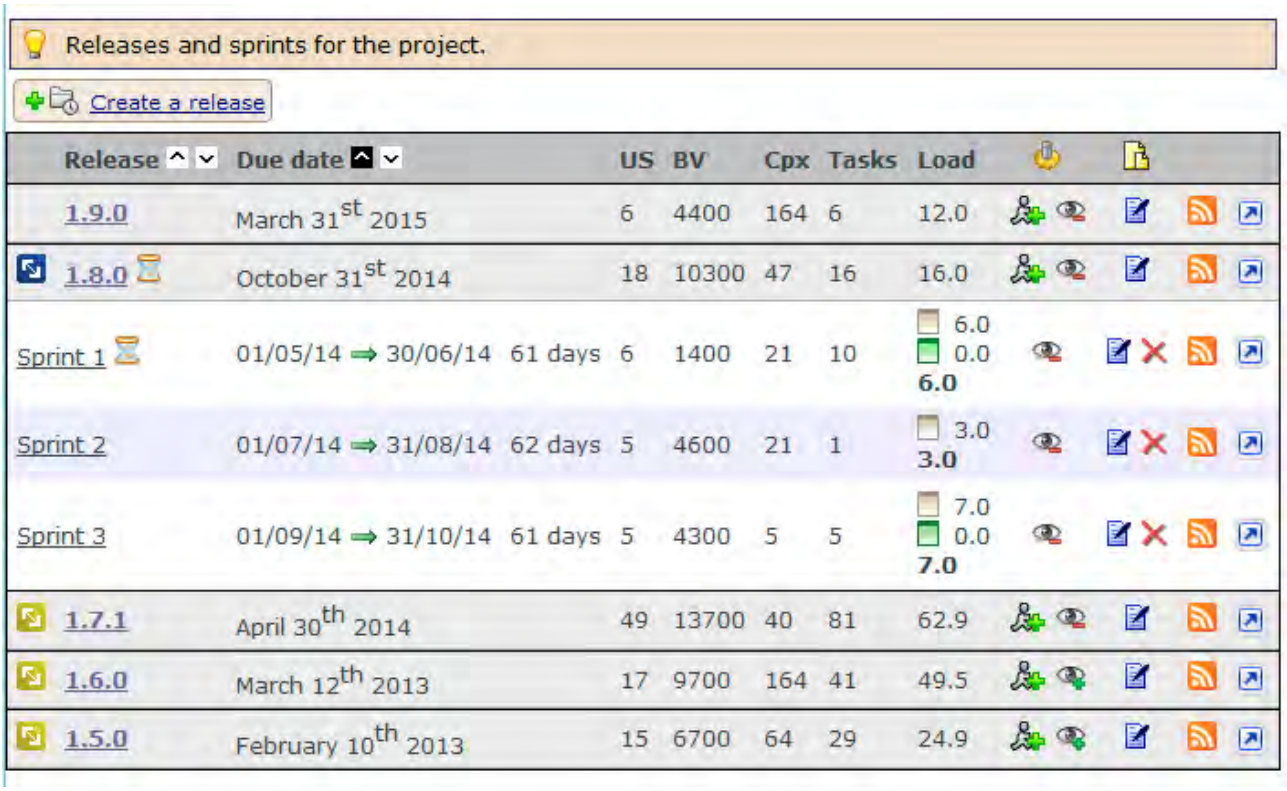
KADOS proposes to create three types of projects depending on the levels of complexity you want to manage:

- one level for a simple dashboard with one line by User Story and any tasks for each User Story
- two levels if you want to have releases in your projects: each release will have a dashboard with tasks
- three levels if you have a project with phased releases such as Scrum where releases have sprints.

This feature allows to cover all type of agile projects and even to flirt with Kanban. Let's take a look at the Scrum project that has the complete set of features of the three project types

Scrum Project

A Scrum project is structured with three levels: project, releases and sprints. Each level has its own backlog. These levels can be viewed and created in the project cockpit. All the releases and sprints form a project plan.



Release	Due date		US	BV	Cpx	Tasks	Load
1.9.0	March 31st 2015		6	4400	164	6	12.0
1.8.0	October 31st 2014		18	10300	47	16	16.0
Sprint 1	01/05/14 ⇒ 30/06/14	61 days	6	1400	21	10	6.0 / 0.0 / **6.0**
Sprint 2	01/07/14 ⇒ 31/08/14	62 days	5	4600	21	1	3.0 / **3.0**
Sprint 3	01/09/14 ⇒ 31/10/14	61 days	5	4300	5	5	7.0 / 0.0 / **7.0**
1.7.1	April 30th 2014		49	13700	40	81	62.9
1.6.0	March 12th 2013		17	9700	164	41	49.5
1.5.0	February 10th 2013		15	6700	64	29	24.9

Figure 1. Releases and sprint hierarchy.

The second main screen for a project is what is called a Kanban. It is not strictly speaking a true “Kanban”, it is more a board of post-its that we will call a dashboard. Dashboards are used to update some attributes of the post-its: status, allocation to release or sprints, Business Value, complexity, etc...

In a project, you'll find many types of post-its: User Stories, Tasks, Issues, Actions, Activities... Most of them can have subtypes that are defined with different colors. The main dashboards are used for allocating User Stories to release and sprints following some Scrum activities and for changing Tasks status

The figure 2 below is taken from the KADOS roadmap. You can find yellow post-its for real User Stories (feature request) and pink post-its for Bugs found by users. These post-its are broken down into tasks. The figure 3 shows light brown post-its linked to each User Story or Bug, describing development tasks. Different colors can be added to have different types of tasks.

Many KADOS users (for Scrum projects) have one color for development tasks, one for bugs and another one for tests. Thus you can describe and manage entirely a User Story without any other tool, setting KADOS as the Scrum team tool.

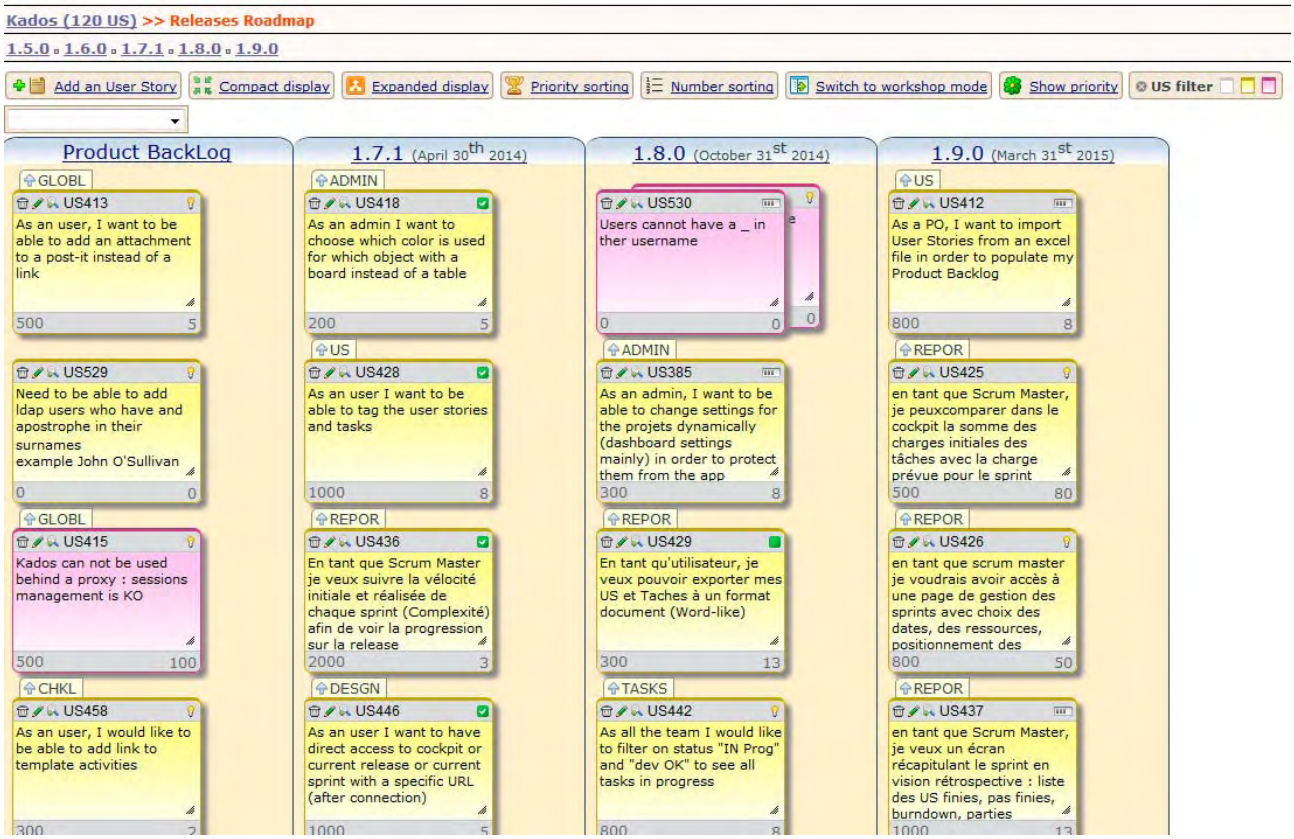


Figure 2: Product backlog and allocation of User Stories by release

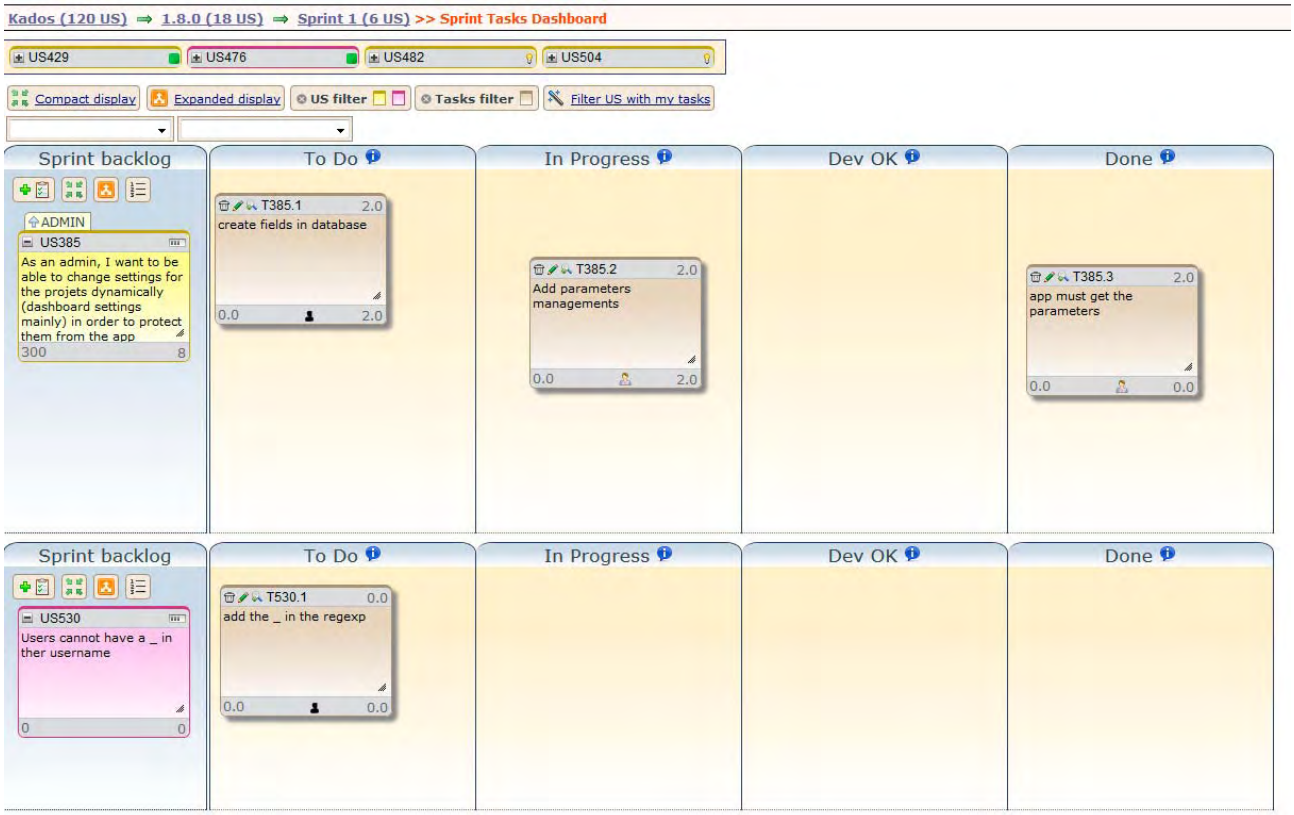


Figure 3. Sprint backlog and Tasks Status Board

Others features

LDAP

By setting a parameter and adding LDAP configuration, you can add users from a LDAP mixed with local users. KADOS will try to find users in its database first and if the search is not successful, it will try to find the user in LDAP. It will then create the user in its database (without the password)

Risk and Problems Management

Each project has an “Issues” board allowing users to manage Risks, Problems and Actions plans. Risks and Problems can be linked to all post-its of the User Stories type.

Features Management

Features are used to group User Stories in any functions or any meta-description that you have in your project. User Stories are allocated to features with a visual dashboard (What a surprise!).

External connections

You may have an external project management tool that cannot be replaced by KADOS (and that is not KADOS job). You will be able to create a project from that tool, by configuring access to the project management tool database, but only if it is a MySQL database. One day, there will be web-services, be patient!

Tags

Tags are small post-its that can be stuck on each standard post-it. They have a color and a very short name and are displayed on the right side of the post-it. They are useful to specify to which transversal topic the User Story is linked or to add any taxonomy on post-its for instance.

Colors Management

Colors are used for all post-its and tags in order to set subtypes for types of post-its and to differentiate tags. They are first created in a color list and allocated after that to post-its with a visual dashboard. You have been warned before, visual board is my hobby!

RSS

The progress of User Stories can be followed with RSS. Just use a link depending on the scope you want to follow: project, release or sprint.

Check-list

To manage project activities that are not directly related to project content, you may use a checklist of activities which is a very simple visual dashboard with three columns (ToDo, In progress and Done). There is one checklist by release. It can be used to follow documents writing activities or relationships with other projects...

Roadmap

KADOS has a roadmap, letting users to push new features requests or bugs to the development team. It is, of course, managed in a KADOS project!

KADOS features are quite mature since the 1.6 release, but there is a lot of new requests and work for a few years!

Conclusion

KADOS is an agile project management tool focused on providing maximum visual management capacity to its users, while being usable for many projects or activities out of the world of Agile software development.

Based on widely used technologies (PHP and MySQL), it can be easily installed and used by people outside of the software galaxy. We have heard stories from users that used it to manage a trial and or a Request for Proposal process.

Try it!

References

Web site: http://www.kados.info

Documentation site: http://docs.kados.info

Roadmap: http://roadmap.kados.info (credentials: pokados/pokados)

Demo site: http://demo/kados.info

Download: http://sourceforge.net/projects/kados/files

KADOS on Ohloh: https://www.ohloh.net/p/kados

Facebook page: http://www.facebook.com/kadosTool

Twitter: https://twitter.com/ScrumKados

EnvJasmine - Test Your JavaScript Anywhere

Trevor Lalish-Menagh, http://trevmex.com

EnvJasmine is a test harness designed to allow you to test the JavaScript code in your project without the need of a web browser. EnvJasmine doesn't care what language your application is written in nor which frameworks you use, it is designed to be completely self-contained. It is meant to be easy to drop into your application so your development team can perform automated front-end testing, catch more bugs and produce higher quality code.

Web Site: https://github.com/trevmex/EnvJasmine/
Version tested: 1.7.2
System requirements: Java 5 or greater
License & Pricing: Open Source, MIT License, Free
Support: Issues can be submitted at the issue section of the web site: https://github.com/trevmex/EnvJasmine/issues.

Introduction

When we first wrote EnvJasmine in 2011, testing your JavaScript code was somewhat of a foreign concept in the world of enterprise web applications. JavaScript testing suites were few and far between. The only developers you could find using them were library maintainers [1] and radically novel consulting firms [2]. There were test suites like QUnit and Jasmine, but they were all designed to run within a browser. At the time I was working on a large Java web application that was being auto-run by our Hudson [3] continuous integration server every time someone checked code into our source control system. What I was looking for was a way to write JavaScript unit tests that would run on every check-in and "break the build" if a test failed, just like the Java developers had on the team. We also thought it would be important to make a tool that would be usable by not only Java web projects, but any project regardless of language or framework, and that is exactly what EnvJasmine delivers.

EnvJasmine is built off of two core technologies: EnvJS [4] - a headless browser written completely in JavaScript and Jasmine - a test framework inspired by RSpec [5]. With EnvJasmine, developers write JavaScript tests in Jasmine that are then loaded in the EnvJS headless browser instead of a normal web browser, decoupling your test environment from external resources. The results are produced as text with a proper exit status code (0 for passing, 1 for failure) to allow continuous integration servers to break the build.

Why "breaking the build" matters

To someone outside the industry, breaking something, especially a build, sounds like a horrendously bad thing. I would argue, though, that it is just the opposite. What I mean by breaking the build is having the continuous integration server fail when it encounters an error. That might seem fairly obvious, but it is astonishing how many projects today, especially in the enterprise space, do not have this ability whether due to lack of infrastructure, lack of planning or both.

The Java community has had this concept baked into it for years with test frameworks like JUnit [6] tying into auto-runners like Maven [7] that fail to compile if the tests do not pass. JavaScript, had been left in the dust until very recently. What EnvJasmine allows you to do is write tests in JavaScript for JavaScript that forces your project to fail to compile if the tests do not pass. This is a very good thing. Before the advent of EnvJasmine and other tools that came along later [8],

even if you did write automated JavaScript tests they would have to be run by hand on a computer with a web browser. Most continuous integration servers are not set up to tie into a web browser [9] so oftentimes the tests would not be run at all. Pushing the JavaScript testing farther down the development lifecycle allows for more defects to be introduced into the code. When you break the build, it stops the defects from ever being released in the first place since the tests break at the earliest possible point (at code check-in). Early detection equals higher quality.

Getting Started

First, you will want to download EnvJasmine from our GitHub repository [10]. In your project's test directory (if you do not have a test directory this would be a good time to create one, call it "tests") create a "javascripts" directory and unzip EnvJasmine there.

On the command line, run the following command from your newly created "tests/javascripts" directory: `bin/run_all_tests.sh` (on Mac or UNIX) or `bin\run_all_tests.bat` (on Windows). You should see the results of the sample tests pass on your screen. From here you can follow the section of the web site entitled "A Basic Tutorial" to get yourself up and running with writing EnvJasmine tests.

Your team will write the tests themselves in Jasmine. The Jasmine format is similar to RSpec and very easy to pick up. The Jasmine documentation page [11] is a great place to start for learning how to write Jasmine tests. EnvJasmine has Jasmine jQuery [12] and Jasmine AJAX [13] built into it to allow you to write test for your jQuery and AJAX JavaScript code. This means that if you are using jQuery, you can write tests that utilize special jQuery function and anonymous callback functions in AJAX calls will be testable as well.

Integration with Maven and Ruby on Rails

Because Maven and Ruby on Rails [14] are such common frameworks, we have created some example projects that integrate EnvJasmine into them. The Maven example [15] adds a plugin into your pom.xml file [16] that triggers all the tests to be run automatically whenever the test phase is executed. The Ruby on Rails example [17] adds a `test:javascript` rake [18] task in the lib/tasks/env_jasmine.rake file [19] that is executed whenever the test rake task is run. If you are using these tools, I encourage you to take a look at the example projects to get started.

Integration with SonarQube for Code Coverage

Code coverage tools allow you to have a visual representation of which lines of code you have written tests for and which lines still need tests. High code coverage helps your developers catch the parts of their code they forgot to test. EnvJasmine integrates with SonarQube [20] to give you code coverage statics on your project.

SonarQube is a fantastic tool for doing deep code analysis in your project. If you are fortunate enough to have this tool set up in your project, it is easy to integrate EnvJasmine into your SonarQube reports to see JavaScript code coverage. The instructions can be found on the EnvJasmine web site in the JSCover section [21].

Introducing JavaScript testing to your team

The technical hurdles of setting up EnvJasmine are small compared to the hurdles of changing your team to embrace a culture of testing. Having done this on a number of teams, I have found

there are a few concerns that come up time and time again. As the change agent on your team advocating for JavaScript testing and breaking the build, you have to make sure you frame the experience in a positive light. You will get a lot of resistance to the idea, especially if your team is not used to developer-driven unit testing. Some of the common complaints are “writing tests will make it take longer to get things done,” “I don’t have time to learn a new framework (Jasmine)” and “that is QA’s job.” It is your job to answer these issues and stay positive. Let’s look at these three complaints a bit closer.

“Writing tests will make it take longer to get things done.” Initially this is true. For the developer, it will take longer, but we have to consider the developer only one part of the project team. Automated tests mean less work for QA and less bugs being released to production. A developer spends more time fixing a bug that escaped to production than they do writing a test up-front. Writing tests in the long-term safe time in the future by catching defects that would otherwise be missed and have to be fixed later.

“I don’t have time to learn a new framework.” There are couple of ways to approach this one: from the top and from the bottom. From the top, you can get management to buy into your idea that testing is good, therefore they should allow your team time to train on testing tools like Jasmine. This is ideal, since it will get the more reluctant among your development team motivated to test. From the bottom, you can study Jasmine yourself, give lunchtime training sessions to developers and make yourself available to field questions from your fellow developers when they are getting started. In truth, both these approaches in tandem make for a very successful testing adoption rate.

“That is QA’s job.” This is an old traditional waterfall response to the idea that developers should write tests that still prevails in many enterprise settings. My main argument here is that the developer while writing their own code is the one person that knows there code the best, therefore is the best person to take (at least) a first pass at writing tests for the code. Automated testing using tools like EnvJasmine are **not** a replacement for a quality assurance team. QA engineers think about your product in a different way than developers, and that insight allows for more robust testing, but the developer, being closest to the code should be the one exercising the logic of the code to proof that it does what they expect it to do.

Contributing

Give EnvJasmine a try, and open up an issue on the GitHub repository for assistance. We are also very open to pull requests, so if there is something you want EnvJasmine to do for you that it doesn’t, we encourage you to contribute. It takes a village to make a great development tool.

JavaScript Testing Today

As I mentioned at the beginning of the article, in 2011 JavaScript testing was in its infancy. Today in 2014, testing your JavaScript code is a well-established norm among small and medium-sized web development companies, and is slowly making inroads in large enterprise teams around the world. EnvJasmine is a great tool for integrating JavaScript testing into an older system, a large enterprise system or a Greenfield project. In this time of hyper awareness on code quality, there is no reason **not** to ensure your JavaScript is well-tested and can break the build. With any luck, EnvJasmine can help get you there.

References

1. For example, jQuery's QUnit (http://qunitjs.com) had been around for a while at this point.
2. Pivotal Labs, inventor of Jasmine (http://jasmine.github.io), which is used in EnvJasmine.
3. http://hudson-ci.org
4. http://www.envjs.com
5. http://www.methodsandtools.com/tools/rspecbestpractices.php
6. http://junit.org
7. https://maven.apache.org
8. Teaspoon (https://github.com/modeset/teaspoon) is a great one for Ruby on Rails projects, but it is not cross-language or cross-framework.
9. Although it is possible nowadays with tools like Selenium Grid (http://docs.seleniumhq.org).
10. https://github.com/trevmex/EnvJasmine/archive/master.zip
11. http://jasmine.github.io/1.3/introduction.html
12. https://github.com/velesin/jasmine-jquery
13. https://github.com/pivotal/jasmine-ajax
14. http://rubyonrails.org
15. https://github.com/trevmex/EnvJasmine-Maven-Example
16. https://github.com/trevmex/EnvJasmine-Maven-Example/blob/master/pom.xml
17. https://github.com/trevmex/EnvJasmine-Rails3-Example
18. http://rake.rubyforge.org
19. https://github.com/trevmex/EnvJasmine-Rails3-Example/blob/master/lib/tasks/env_jasmine.rake
20. http://www.sonarqube.org
21. https://github.com/trevmex/EnvJasmine/blob/master/lib/jscover/README.textile

Cuke_sniffer - Static Analysis for Cucumber

Robert Cochran, @cochrarj

cuke_sniffer is a free open source static analysis tool for Cucumber that identifies smells and issues in a project. Cucumber is a tool that executes plain-text functional descriptions as automated tests in a behavior-driven development (BDD) style.

Web Site: https://github.com/r-cochran/cuke_sniffer
Version tested: 0.0.8
System Requirements: Windows, OSX, Linux with Ruby 1.9.3 or later
License & Pricing: Free, MIT license
Support: Issue tracker and wiki at https://github.com/r-cochran/cuke_sniffer.

Cucumber was introduced to the software profession as a way for teams to explore and refine their practice of Behavior Driven Development(BDD). Product Owners, Developers, and Testers are given the flexibility to discuss what the system needs to do in business terms in the context of the problem being solved. However, too often teams see Cucumber as an automation solution and treat it as such; creating scenarios that are long and drawn out with reused steps instead of short and concise stories. As much as we value Cucumber as a communications tool to bring the team and business together, it is still code and should be held to a quality standard amongst the team.

cuke_sniffer was born out of the frustration of dealing with unhealthy Cucumber projects that are hard to read and maintain. The ultimate goals are to educate and to empower people who want to use Cucumber. In order to understand the problem we have to see how it happened and think about what could have been done better. Was it the junior team member who didn't know better? Was it the team member who only cared about a passing test and started to copy and paste everything? Was it me in my learning of a feature in Cucumber and over using it, just to feel the pain of it later? (I'm looking at you nested step definitions).

With the freedom to communicate anything in a story, people tend to paint themselves into a corner of anti-patterns and abuse. Like any code base, Cucumber projects can accumulate technical debt and begin to rot. Some projects just build on top of that rot with little consideration of the difficulties they will have to answer to in the future.

The concepts of using Cucumber are simple enough. You have a Feature file that has scenarios that tell a story in a given/when/then order which is linked to code that tests the application. Here is an example of what you might find in a basic introduction to Cucumber:

```
Feature: Buying a sandwich

Scenario: As a customer I can buy a sandwich with my credit card
  Given I have ordered a sandwich
  When I pay with my credit card
  Then I get my sandwich
```

Cucumber can be pretty simple to use. Now, anyone who has been using Cucumber in an enterprise or team environment can tell you that the Scenarios don't always stay this simple. The business now needs to specify what credit cards can be accepted, what sides can be sold with the sandwich, and what drinks are offered.

It would be nice to identify all the scenarios when the feature is written but these enhancements will happen over the life of the project. Your nice feature for buying a sandwich with a credit card might transform into something like this:

```
Feature: Buying a sandwich

@wip
Scenario Outline: As a customer I can buy a sandwich with my credit card
  #Given I have ordered a sandwich
  When I have ordered a grilled cheese sandwich
  And select a root beer from the drink drop down
  And check the pickle check box
  And chilli cheese fries
  #When I pay with my credit card
  Given I pay with my <card_name> credit card
  When I click the submit button
  Then I get my sandwich
  And the card is billed automatically
Examples:
  | card_name .....| #1
  | Visa.....| #2
  | Master Card .....| #3
  | American Express .....| #4
  #| Debit .....| #5
  ...
  | Discover.....| #25
```

It's not pretty and as much as we might not like to believe it, there is probably a scenario like this in our project. We might be able to forgive that it is *clearly* a work in progress, but the @wip tag is often forgotten when the scenario is completed. The scenario is out of the Given/When/Then order. There are entirely too many steps to manage; if a scenario is more than 3-5 steps long it is too complex or can be more concise. There are steps that are commented out and serve no purpose. Our scenarios should tell a story and not specify directly what the system needs to look like or react to. Implementation words like drop down, check box, and button distract from the story. The examples table is way too large and for some reason Debit is no longer a valid example. Without the review of our Cucumber projects we might simply miss or ignore these problems.

cuke_sniffer is my attempt to automate the feedback and identification of these issues as well as give a representation of the size and health of a Cucumber project. To help quantify the health of the various parts of Cucumber I chose to use a golf like scoring system, more points mean more areas to improve. Unlike golf, the score of a project is a means to monitor growth and improvement of a project and really can't be compared against others.

When cuke_sniffer is used against the ugly example we are automatically informed about several of the issues I mentioned before and more. cuke_sniffer will give a description of the issue found as well as the number of times it was found.

Improvements to make:
(2) Commented step.
(1) Scenario Outline with too many examples.
(1) Implementation word used: drop down.
(1) Given/When/Then used multiple times in the same Scenario.
(1) Implementation word used: click.
(1) Scenario steps out of Given/When/Then order.
(1) Scenario with too many steps.
(1) Commented example.
(1) Implementation word used: button.
(1) Tag appears on all scenarios.

With this information about our scenario we can begin to identify what needs the most improvement. We can educate ourselves on why the rule fired and how to prevent causing more smells in the project. With just knowing about the problem and being mindful not to cause it to continue you have given yourself a fighting chance of cleaning up the project. The improvements are identified when cuke_sniffer goes over your Features/Scenarios/Step Definitions/Hooks files and executes rules against them.

Rules are divided up into 4 categories: information, warning, error, and fatal. Information rules are things that are minor issues that can be situational from team to team and are meant only as advice for improvement of the projects use of Cucumber. Warnings are issues that impact the readability or the usability for parts of your cucumber project. Errors are meant to be a way to help defend projects from debugging problems that are difficult to catch. Lastly, Fatal issues are things that will prevent your project from even executing. The full list of rules can be found on the cuke_sniffer github wiki: https://github.com/r-cochran/cuke_sniffer/wiki/Rules-list.

cuke_sniffer allows for the customization of these rules. If you don't agree with a rule, it can simply be turned off. Maybe the phrasing of the rule doesn't make sense and you want to rewrite it. All components of a rule are customizable. If you think of a new rule that is not found in cuke_sniffer you are able to write the rule and include it. If you do find yourself writing rules let the cuke_sniffer team know and we can include it in future releases to share your work with others! For more details on customizing or writing rules consult the cuke_sniffer github wiki: https://github.com/r-cochran/cuke_sniffer/wiki/Customizing-Rules.

Just because a rule fired against a Scenario does not automatically mean it is bad. cuke_sniffer specifies a customizable threshold of points that must be met for a Cucumber object to be considered bad. Customizing this threshold allows for you to slowly slide the bar up or down for your Cucumber objects. At the start of improving your Cucumber project you might slide the threshold higher to set a goal of making all Cucumber objects good. As time goes on you might slide the threshold down to continue doing improvements.

When cuke_sniffer is ran it will give you the list of improvements as well as the score but it will also give you a breakdown of the various Cucumber objects. You will learn about what your worst and best scores are as well as the average of your Cucumber objects. Additionally, you will be given an idea of what percentage of your Cucumber objects are good or bad.

Cuke Sniffer

Score: 11764

Summary -

	Features	Scenarios	Step Definitions	Hooks
Total Score (?)	180	11450	58	76
Count (?)	4	19	8	5
Lowest Score (?)	20	10	3	0
Highest Score (?)	80	5585	21	26
Average Score (?)	45.0	602.63	7.25	15.2
Threshold (?)	30	30	20	20
Good (?)	50.0 %	84.21 %	87.5 %	40.0 %
Bad (?)	50.0 %	15.79 %	12.5 %	60.0 %

While Features and Scenarios are the most visible and problematic objects in Cucumber, Step Definitions are where we spend most of our time and they have unique problems. Steps Definitions have the inherit intention of being reused for multiple scenarios. If I am describing how a certain feature works with several scenarios I am most likely going to reuse my setup(Given) or action(When) steps. However, as a developer I will want to reuse any existing Step Definitions. Suppose I find a Step Definition that I did not write that looks like it matches exactly what I want to do. *Hooray! I don't have to do more work!* I run my test and find that it fails on the step that I reused.

Now I have a problem, I have a Step Definition that doesn't work for my use case and I am left with several options. I could just not use it and write an entire new Step Definition from scrap or I could spend the time to debug the existing Step Definition and hope that my changes don't break anything else. The real reason for the Step Definition not working could be that it never worked or that it was replaced with another Step Definition. It would be very nice if the Step Definition was removed after no one was using it anymore.

cuke_sniffer will catalog all Step Definition calls in a project and identify any that would be considered dead. With this information we can now quickly remove or spot check the Step Definition. By removing the dead Step Definitions we greatly reduce the amount of noise in our Step Definition files and prevent the same scene to play out again. If the decision is to debug the Step Definition we can now have confidence that the changes will not break other tests.

The first time you look at the cuke_sniffer results for your project you might feel overwhelmed. Remember that you don't have to fix everything at once; incremental change and understanding of the problem can often be better than just fixing the problem.

Understand why an improvement would be suggested and try and not to cause more of that issue to occur. Take a report from the start of the week and compare it to a report from the end of the week. Did the total for that improvement go up? Find areas where the improvement is suggested and make the fixes one at a time. Before too long you will be happier with the health of your Cucumber project.

STARWEST, October 12–17 2014, Disneyland Hotel, Anaheim CA

Experience just the right learning formula at STARWEST. The conference includes access to more than 75 learning sessions over six days—all in one convenient location. Exceptional world-renowned keynote speakers such as Julie Gardiner, Bob Galen, Paco Hope, Simon Stewart, and Lee Copeland have been selected to inspire and motivate you. Join us in Anaheim to see these keynotes, expand your peer network, and meet new contacts throughout the week. Register by August 15, and you'll save up to $600 when you use code SW14MT!

Go to http://vlt.me/mtsw14pet

How to Choose a Test Management Tool

While it might seem that the most important consideration when choosing a test management software tool is the set of basic tool features supporting the test process itself, you should not neglect a wide range of other questions that could make or break your test management tool choice. This whitepaper identifies the key attributes you should look for in a test management solution.

Go to http://www.inflectra.com/Ideas/Whitepaper/How-to-choose-a-Test-Management-Tool.aspx

---

Load Testing Tool Brings Enterprise Selenium/WebDriver Features to Non-Coders

Web Performance Tester gives you enterprise-grade web functional and load testing without coding, all in a full supported professional product that's configured and ready to work right out of the box. Your testing team can leverage existing skills while professional training and support are only a phone call away.

Start your free trial now:
http://www.webperformance.com/download/

METHODS & TOOLS is published by **Martinig & Associates,** Rue des Marronniers 25,
CH-1800 Vevey, Switzerland Tel. +41 21 922 13 00 Fax +41 21 921 23 53 www.martinig.ch
Editor: Franco Martinig ISSN 1661-402X
Free subscription on : http://www.methodsandtools.com/forms/submt.php
The content of this publication cannot be reproduced without prior written consent of the publisher
Copyright © 2014, Martinig & Associates