
METHODS & TOOLS

Practical knowledge for the software developer, tester and project manager

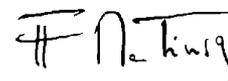
ISSN 1661-402X

Spring 2011 (Volume 19 - number 1)

www.methodsandtools.com

What is a Successful Project?

Recently somebody asked on a forum "when is Scrum not working?" This lead me to the question "when is a project successful?" Traditionally, you can measure the success of project by checking the respect of scope, schedule and budget. In this situation, the burden is more on project managers and developers: they have to estimate the user requests and deliver on these estimates. This requires the mostly complete definition of the requirements at the beginning of project. Every change is the subject to negotiation as it could cause a change in either budget or schedule. In this situation, people might be more judging the capacity of people to estimate... or better to protect themselves from unexpected problems. A more business-oriented definition is to focus on the delivery of value, the faster the better. In this case, the success of a project could be assessed only after delivering the application, computing its return on investment (ROI). The value might be also implicitly present in the traditional project approach, but then the mindset is more strictly focused on the project management results than the value creation output. In practice, value is often not really considered in software development approaches. You rarely see estimates for the value of the feature. The user story format "as a... I want... so that I can...." could add a "for a value of..." section that could improve the visibility of a project. These estimates should be checked with the reality 3,6 or 12 months after delivery. I am sure that not many of us have witnessed this type of validation in their company. To be honest, this is also because it could be difficult to distinguish if an increase in sales is due to the revamping of a web site or a clever marketing campaign. There are also many ways to define the value of a project. Apart financial points, there is also an important human factor in the success of software development projects. Project managers know that it better to optimize value than trying to maximize it. To get the approval of all stakeholders, you might include in the scope some content with limited business value but considered as a "pet feature" by a powerful manager. In large organizations, to ensure end-user acceptance and avoid that the software will be used as the scapegoat for fighting organizational change, you have define the scope around what could be accepted by employees and not only on what you think could deliver the best value. All these practical issue with defining value will make it always easier to judge a project on its own "hard" budget and schedule numbers. This shouldn't prevent to introduce the concept of value during its planning phase, especially when you have to discuss the scope of the project and in which order requirements should implemented.



Content

Dialogue Sheets for Retrospectives and Beyond	page 3
Using Models and Standards.....	page 8
The Psychology of UX.....	page 21
HTML5 for Rich Web Enterprise Applications.....	page 35
Gradle.....	page 41
Saros.....	page 44
StarUML	page 48

Manage Complex Systems with PTC Integrity - Click on ad to reach advertiser web site



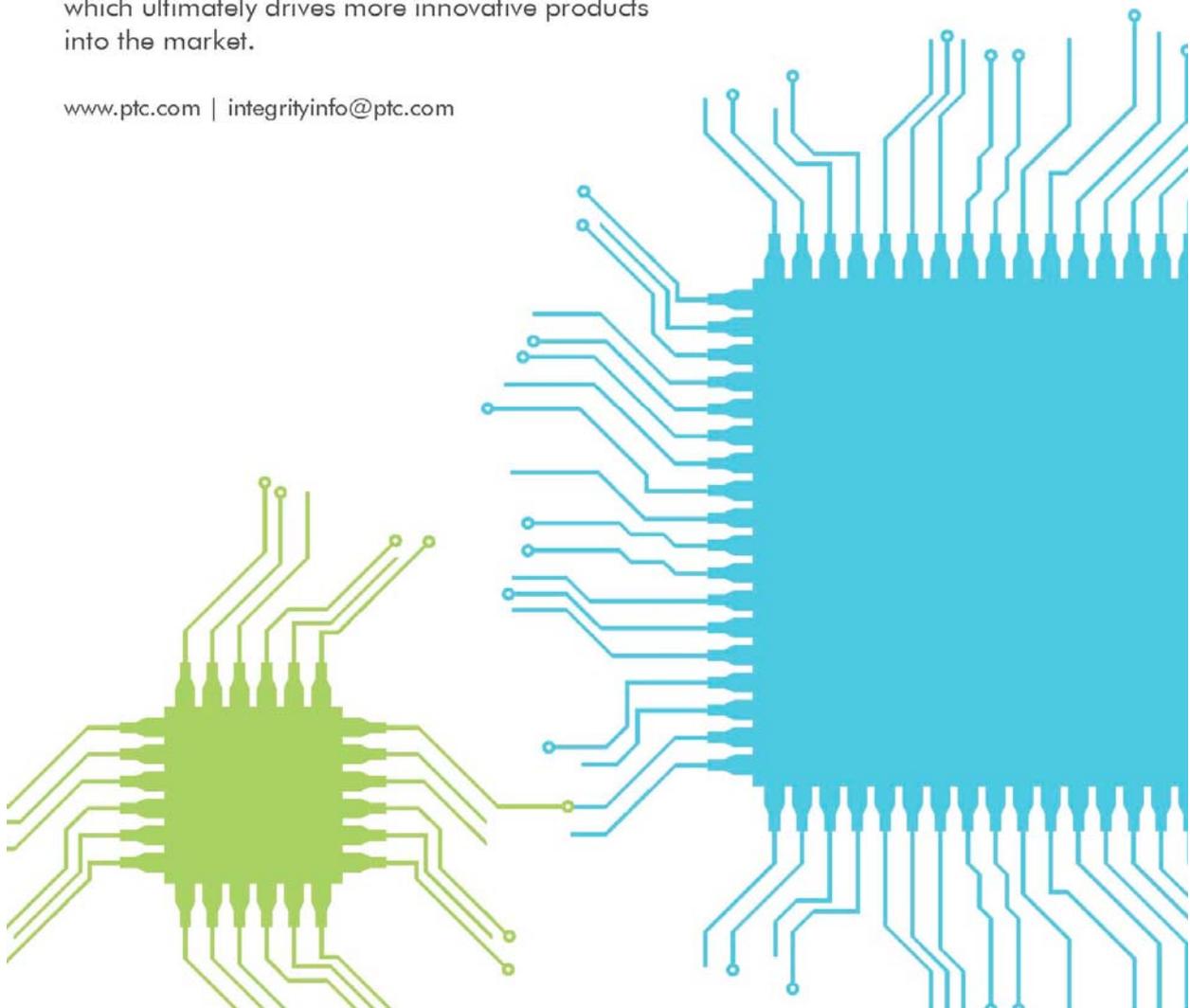
PTC®

Integrity
A PTC Product

Where Software is critical, innovate with Integrity

With Integrity, a PTC product, engineering teams improve productivity and quality, streamline compliance and gain complete product visibility, which ultimately drives more innovative products into the market.

www.ptc.com | integrityinfo@ptc.com



Dialogue Sheets for Retrospectives and Beyond

Allan Kelly, allan @ allankelly.net
Software Strategy Ltd, <http://blog.allankelly.net>

In the last few months I have been trialling a new technique for team retrospectives. This technique involves a large sheet of paper called a "Dialogue Sheet". So far the teams who have tried using a Retrospective Dialogue Sheet have liked them, feedback so far suggests they create good discussion and teamwork. An example of Retrospective Dialogue Sheet is shown in the figure below.

Comments such as "generated intensive & focused discussion" and "plenty of discussion, valid action points" are typical. Not only do the sheets work as a retrospective, they help with team building and can encourage some quieter team members to voice their thoughts.

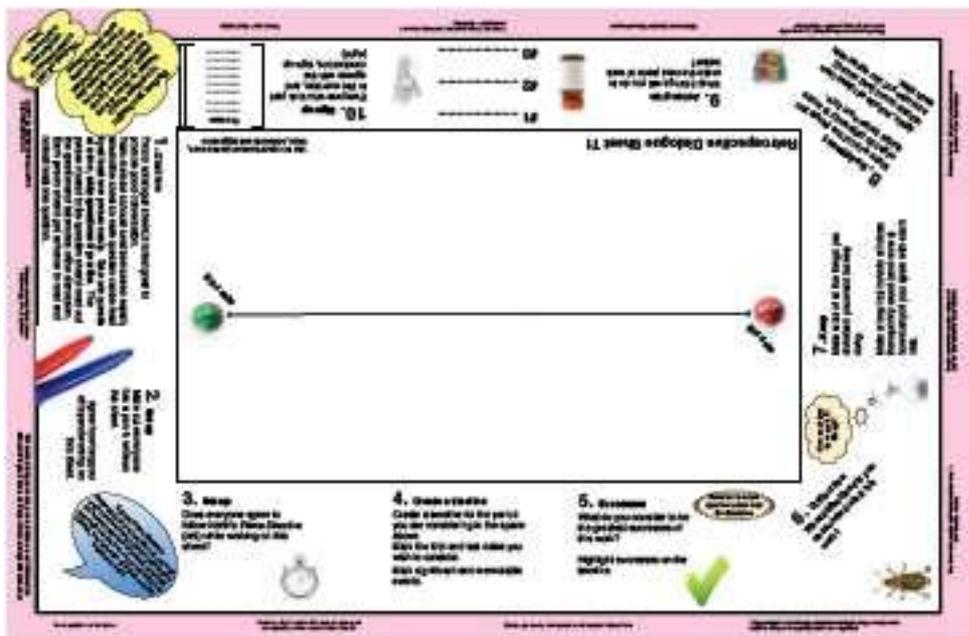


Figure 1 - Example retrospective dialogue sheet

Dialogue sheets

Dialogue sheets are a technique invented in Stockholm at the Royal Institute for Technology, KTH. The intention is to promote good conversation and the method can be used for a variety of discussion-based activities. As far as I know, dialogue sheets have not previously been used for retrospective exercises, or elsewhere in software development. Having taken part in a dialogue sheet based exercise and used them in one of my own workshops it seemed obvious they could be used for retrospectives.

A dialogue sheet itself is a large sheet of paper, A1 or similar size, which is decorated with questions, quotes, illustrations and space for making notes. The questions on the sheet serve to guide the team through the exercise and focus thinking on the subject of the dialogue sheet.

Typically a team of three to eight will spend between one and two hours working the sheet. The team seat themselves around the dialogue sheet - shown in Figure 2 - and take turns asking the questions, facilitating the discussion and making notes on the sheet itself.

Seapine Software Software Quality DNA - Click on ad to reach advertiser web site



We Live Quality

Software quality is in our DNA. For over 15 years, we've lived and breathed it. The reason is simple: Your software affects our friends, our families, and ourselves.

Whether it's the latest video game or a secure banking web site or the software that analyzes medical test results, we want it to work right because we rely on it.

From our award-winning application lifecycle management (ALM) solutions, to our helpful consulting and Agile services teams, to our world-class customer support, Seapine has helped thousands of companies worldwide build, test, and deploy quality software.

Go with Seapine, and get serious about software quality.

www.seapine.com



© 2011 Seapine Software, Inc. All rights reserved.

Three is the lower boundary for the exercise simply because you need three people to have a discussion. Eight seems to be the natural upper limit simply because the sheet has four sides and seating more than two people per side becomes difficult. That said, if seating can be arranged then 10, 11 or 12 people could take part, although this would mean the exercise would need more time.



Figure 2 -- A team working on a dialogue sheet

The team normally records their thoughts, comments and observations directly on to the sheet. At the end of the exercise the sheet may be retained for future reference, hung on a wall or photographed as a record of the exercise and results.

Larger teams are advised to split into several smaller groups each with their own dialogue sheet. When all teams have completed their sheet the teams take turns at examining each others sheets and discussing the results.

In one exercise I split a large team into a group of developers and a group of non-coders (team manager, business analyst, project manager, etc.). The compare and contrast exercise was enlightening: some common problems emerged but also differences in views and understanding.

Time line and beyond

Having sat themselves around the dialogue sheet - with a supply of pens - the team starts the exercise. Each dialogue sheet contains approximately ten questions or instructions. Starting with number one the team takes it in turns to read the question, facilitate the discussion and make notes on the sheet.

The first few questions are actually instructions. These set up the exercise. The next set of questions asks the team to think about the work they have undertaken and how it could be better. These questions start by asking the team to complete a timeline in the middle of the sheet. Teams may discuss any time period they wish, most choose to focus on the last sprint or release is the usual choice. Having decided the start and end of the timeline all team members help mark memorable events on the timeline - successes, failures, ups, down, fun, sorrow or anything else, the aim to reawaken memories of the whole period.

So far the retrospective dialogue sheets have all used a timeline in the middle of the sheet as the focus for the exercise. The intention is to create some sheets using alternative exercises and focused on particular aspects of software development or problems a team encounters.

Next teams are asked about the successes and difficulties on the work. Different dialogue sheets pose these questions in different ways. The aim here is avoid teams giving the same answers each retrospectives and to encourage them to look at the work in different ways.

The next set of questions encourages the team to think about what they would, and would not, change about the way they have been working. Again different dialogue sheets phrase these questions differently to bring out alternative responses. Throughout the process the team are encouraged to talk and discuss. The reader of the question, and thus the temporary facilitator and recorder, is changing regularly, so everyone gets a chance to speak. Indeed, this encourages those who normally shy away from talking to become involved.

The closing questions aim to bring the team to a conclusion and agree on what they will do differently next time to make things better. As far as possible the team is encouraged to come up with concrete actions. Not just "improve communication" but describe what they will actually do to improve communication.

Pictures decorate the sheet and around the edge are quotations, both aim to be thought provoking and inspire the team to reflect on the nature of their work. The quotations play no formal part in process and tend to work on the individual's thought process.

The facilitator

One of my objectives in creating the retrospectives dialogue sheets was to remove the role of facilitator. Not that I have anything against retrospective facilitators - I've facilitated a quite a few myself - but I noticed several reoccurring situations where the facilitator might inhibit discussion. The most obvious situation is where there is simply no facilitator available. Without a facilitator the retrospective either doesn't happen or someone who is not ideal takes over.

There are a number of people who naturally step into the facilitator role and who might not be the right person to do so. This could be someone with fixed opinions who expects a certain outcome, or it might be someone who holds a position of authority in a team - a Project Manager or Architect.

SpiraTeam, the Complete Agile Solution - Click on ad to reach advertiser web site



Used by over 800 customers worldwide...

SpiraTeam® - The Complete Agile Solution!

Tired of having to deal with separate tools for your Agile development and testing activities?

SpiraTeam® provides a complete Agile Lifecycle Management solution that handles your User Stories, Acceptance Tests, Sprint Plans, Bugs and Issues in one integrated environment. **Stop making things hard for yourself!**

> [Learn More](#) | [Download Free Trial](#)

When a Project Manager or Architect takes the facilitator role they face a difficulty: *do they include their views and observations in the retrospective or do they remain silent?* If they speak they compromise the facilitator role, if they don't they don't get their own voice and they don't give team the benefit of their experience.

For team members, having an authority figure conduct the retrospective may inhibit their comments and analysis. It can be hard to say "The project plan was unrealistic" when the project manager who wrote it is facilitating.

Some teams rotate the facilitator role between them or bring in co-workers in different teams to facilitate. However this is not an option for small teams or when there are few other people available.

Removing the facilitator role side-steps all these issues and can lead to new insights. One team commented that having someone facilitate naturally led to a focus on the facilitator, using the dialogue sheet this focus was removed and the team could focus on the retrospective.

Again teams seem divided on whether facilitator-less retrospectives are a good idea. Not all are convinced, some feel facilitators help ensure everyone gets a voice and speaks, others find it opens up the exercise. Either way, for the first time, dialogue sheets offer the option of a facilitator-less retrospective.

What is available?

At the time of writing there are three retrospective dialogue sheets available in "beta test". The intention is to refine these sheets and add some more based on different exercises. The sheets are largely agnostic although they do assume a more Agile approach is desired.

One new sheet is tailored for the end training course retrospectives. I am using this sheet on my own training courses at the moment and it has been well received.

Another request has been for a dialogue sheet based on advancing Agile understanding and practices. Right now I'm testing a *future-spective* version of the sheets for use by new teams to help them agree common working practices and approaches. This sheet should be available by the time you read this.

At the moment the sheets are only available in English. In time I hope to have them translated to other languages.

These dialogue sheets can be downloaded for free from a dedicated mini-site, <http://www.dialoguesheets.com>, once downloaded they need to be printed. This requires either an oversized printer or a visit to a printer shop.

Readers who work at large companies will probably be able to find a large printer in the company, or may have access to a reprographics department who have a large printer. For everyone else a print-on-demand service has been arranged so you can buy printed sheets for a small fee plus postage.

Finally

So far the trials of Retrospective Dialogue Sheets have been very encouraging. On the whole teams like them, there have, naturally, been suggestions for improvement. So far everyone who has used one has said they will use them again. Indeed, at one company a test engineer like the sheets so much she went around enthusing about them to other teams.

Some teams have declined to try a dialogue sheet, these teams feel their existing retrospectives work well. I would encourage these teams to give a dialogue sheet a try. Using one once does not commit using one forever more and they may well lead to new insights or perspectives.

Personally I find the biggest challenge when holding regular retrospective is to keep them fresh and keep new insights coming. Dialogue sheets offer one more tool for varying the retrospective and different dialogue sheets further increases the variety.

Dialogue Sheets for Retrospectives are now a proven retrospective technique, and one with a lot of potential in future not just for retrospectives but for learning generally. While they are not the only retrospective technique they are a new tool in the toolbox.

More information

For more information on dialogue sheets visit <http://www.dialoguesheets.com>.

Using Models and Standards

Sue Rule & P. Grant Rule, s.rule @ smsexemplar.com
Software Measurement Services Ltd., www.smsexemplar.com

Limitations of models

IT is at the heart of an organisation, enabling and supporting everything the organisation does from HR, to product development to customer service. In many areas of industry and public service, IT no longer simply supports other parts of the organisation to develop products and serve customers, IT is integral to the product and the way customers interact with the business.

The challenge for IT professionals is to find effective ways of realising the full potential of technology-intensive business systems. Considerable know-how and experience has been accumulated over IT 50-year business history, and much has been invested in devising quality models and standards. However, many myths also abound, frequently created or promoted by parties with a vested interest in a particular model. Quality standards show you where to improve - but not how or why.

An evidence-driven approach should be taken to the adoption of standards and frameworks, as with every other business decision:

- Understand the model-maker's purpose, and use the guidance provided in appropriate ways. It is *guidance*; not a rulebook.
- Focus on the outcome not the certificate. Compliance, or the attainment of certificates and interim targets, all too often becomes confused with true business goals. This is dysfunctional; meaning it is behaviour which *detracts from* overall business performance.
- There should always be a sound business case for adopting an improvement programme, defining clear business benefits and measurable outcomes.
- Measure progress and *business results* - not compliance to the model. If it matters, measure it. And if it doesn't, don't. It is possible to be effective and compliant, but only if effectiveness is the focus. A focus on compliance will not deliver improved effectiveness - or the associated business benefits of improved effectiveness.
- Communication is key to successful and sustainable improvement. Good measurement practices and short feedback loops are vital.

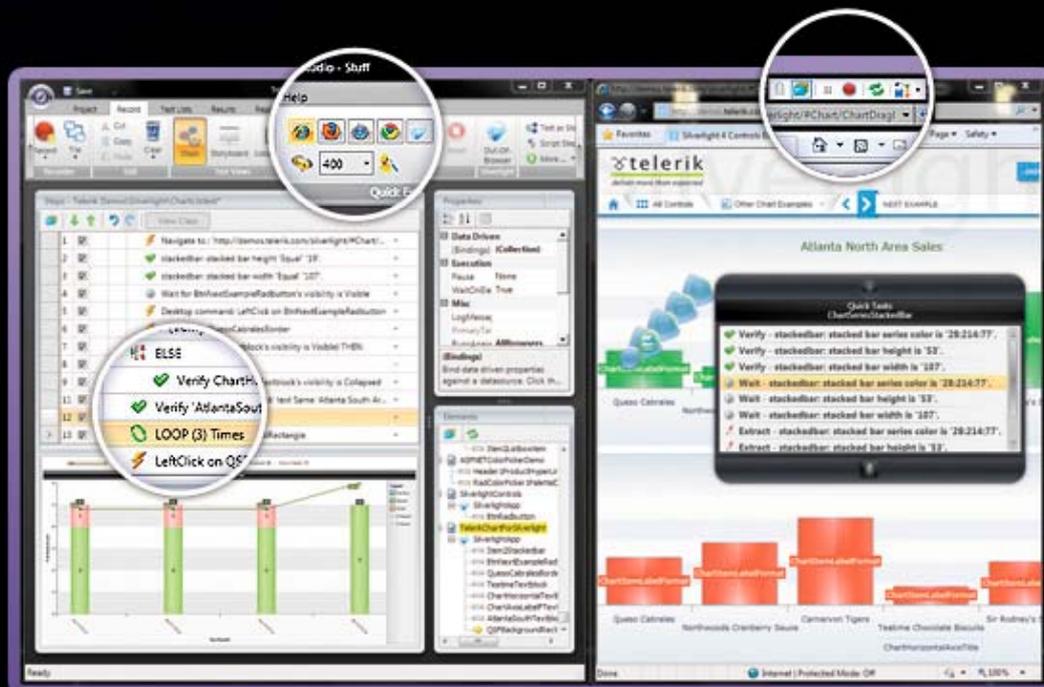
There are many bear-traps into which numerous well-intentioned improvement initiatives have disappeared without trace. Integrated team-working, responsibility-based performance management, and many other behaviours which support value focus, continuous flow, and pull, are often counter-intuitive, especially to staff used to operating in a compliance culture. Because 75-80% of organisations are 'average', achieving typically poor levels of performance, few people ever experience 'high performance' and do not realise there are better ways of doing things than those they are used to. To achieve real changes in effectiveness, most organisations will need a specialist guide whose experience goes beyond knowledge of the models themselves.

A prescriptive approach is not sufficient to move IT business role from cost centre to strategic enabler.

Testing is Now Radically Easier with Telerik Test Studio - Click on ad to reach advertiser web site

Testing is Now Radically Easier

Telerik Test Studio



Easy Test Creation

- Test any app – HTML | AJAX | Silverlight | WPF
- Intuitive point-and-click recording
- Record once, run against multiple browsers

Easy Test Maintenance and Reporting

- Element abstraction and reuse
- Remote scheduling, execution and results reporting
- Seamless QA-Developer collaboration

[DOWNLOAD FREE TRIAL](#)

telerik
deliver more than expected

Used correctly, models and standards can support continuous improvement in the quality of IT services, particularly in organisations starting from a very poor understanding of process performance. They support a more consistent and reliable service to business users, and potentially a more competitive business.

But the benefits all lie in correct implementation. If the focus is on compliance without understanding of the value stream, models can hinder effectiveness and efficiency, not improve it. If those using processes are not empowered to change or improve them, process performance is likely to be a constraint on high business performance. This is why the Agile Manifesto emphasises people over process. People are the creative intelligence of the technology-enabled value stream. They must be empowered to use their intelligence and creativity, not constrained to adapt to prescriptive, overly-bureaucratic processes.

Lean-Agile Systems

Lean service focus is about hearts and minds. It is a change of business culture and behaviours which has far greater impact on the value delivered to customers than any prescriptive process. An informed understanding and application of the principles of lean flow production goes to the heart of the organisation and touches everything the organisation does, including its IT infrastructure.

The modern principles of Lean Thinking are derived from the Toyota Lean Production System, but make use of found truths that can be traced back to ancient civilisations. Numerous historic examples show that two key concepts inform effective systems of work, and these concepts are encapsulated in modern Lean thinking as:

Just In Time: Creating an end-to-end system of production that is so in tune with customer demand it can produce the right product or service for its customers, at the right time, for the right price, first time, every time.

Jidoka: Designing-in quality rather than fixing failures. Jidoka implies that everyone engaged in the value stream is empowered to *stop production* rather than let sub-standard delivery be passed on to the next process step. (Contrast this concept with the exhaustive testing carried out on most software products.)

Lean workflow requires the organisation's software products, and its processes for developing new software products, to be aligned to the delivery of stakeholder value. The products themselves must support a customer-focused service stream most effectively. Agile development methods lend themselves to achieving this alignment. They improve the dialogue between business users and developers, particularly in terms of the collaborative evolution of the software to fulfil real business requirements.

The key to Lean effectiveness however is to change the mindset of software developers to focus on delivering a Lean service. That is, being so in tune with the demands of their business customers that they produce the right product at the right time for the right price, first time, every time. This cannot be achieved without good communications between all the business stakeholders and the software development team, and this remains an issue for many organisations. For businesses at large to benefit fully from the flexibility and responsiveness of Agile development methods, an in-depth knowledge of how software product development delivers value to stakeholders is required. Only by applying such know-how to align the system of work can optimum value be delivered.

Agile methods are still regarded with suspicion by those responsible for corporate procurement and governance. Agile development teams need to recognise the role of process discipline in

delivering value to all the business stakeholders. Process performance should be visible to all stakeholders, particularly in an outsourcing context where trust needs to be established between corporate partners.

The key to high performance in software intensive value streams is to create a Lean Service ethos in the workforce. Coupling good, end-to-end process management with Agile development methods significantly increases organisational effectiveness. The two working together have greater impact than either used alone. Success is delivered by people; failure is generally due to poor process.

Choosing a model

Models capture experience and knowledge that can be applied to develop new products and services, resolve new business issues, or address similar issues occurring in different organisations. Unfortunately, the mere act of capturing experience devalues it. It is rather like an archaeological artefact removed from its site. It is significantly the less without its context and provenance. It is therefore essential to interpret the knowledge contained in a model intelligently and adapt it to the business context in which it is being used.

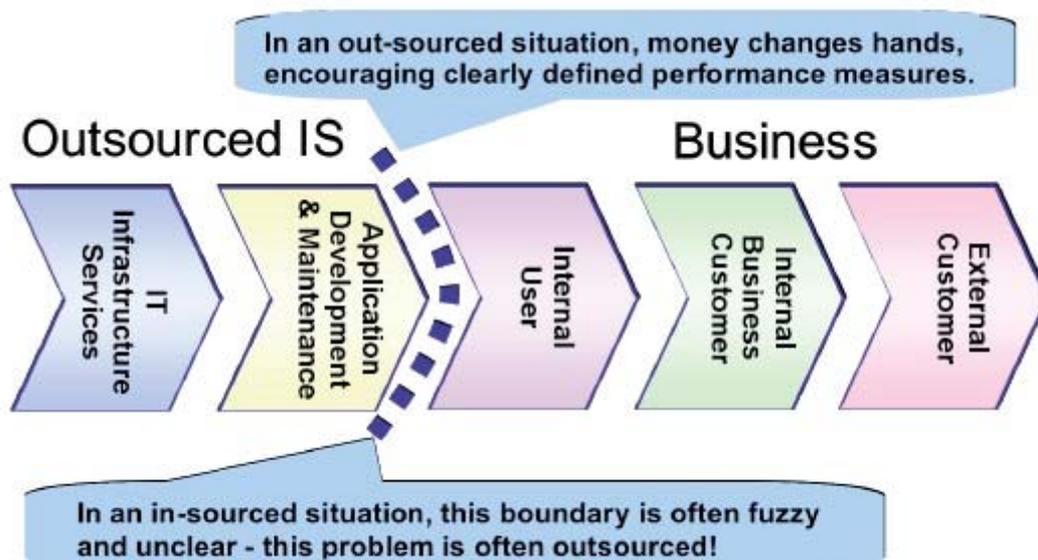


Figure 1. Commercial drivers for the adoption of models and standards

Many models claim to encapsulate ‘best practice’, but experience shows that all models tend to encourage ‘batch and queue’ thinking, with the emphasis on projects rather than on service. This is counter to Lean flow thinking. It is another dysfunctional behaviour pattern.

A model contains only those features that are of primary importance to the model maker's purpose. When choosing a framework to offer guidance to your business, you should ensure you know enough about the model maker's purpose to be confident that it is both **applicable** in your business environment; and **flexible** enough to facilitate the achievement of your business objectives.

Cultural obstacles such as ‘Not Invented Here’ syndrome, resistance to change, and change fatigue can influence decisions relating to the adoption of quality standards, but these should never be allowed to outweigh the business drive for improvement. All these issues can be

overcome by engaging knowledgeable specialist support. It is simply a case of evaluating the risk of continuing ‘business as usual’ against the perceived cost of change. Apart from the business benefits to be derived from improved process performance, there can also be overriding legal and commercial factors in achieving certification against a recognised standard. Much of the interest in quality standards and frameworks has been driven by the rise in outsourced IT services.

By introducing commercial terms to the hand-off between the development team and the business user, the cost of poor process performance is made apparent. Customers seek to ensure good governance by stipulating the attainment of certain standards. Fig. 1 shows the typical break-point in communication between business users and providers of ICT services that models and standards are frequently invoked to address.

However, there are two problems with this approach:

1. Compliance with the model, rather than real improvement in process performance, becomes the goal.
2. Experience shows that process maturity will devolve to the lowest level extant in the end to end value chain; so if a customer with process maturity mapping to Level 1 of the CMMI® model engages with a supplier that has achieved Level 4, the overall maturity level of the supply chain will typically be Level 1.

In an outsourcing partnership, ‘Measures’ - including payments, incentives and contract terms - must be aligned to pull value through the end-to-end process. Measures imperfectly aligned to the creation of customer value incentivise dysfunctional behaviour which will not be addressed by a ‘tick-box’ compliance with a specified standard. An objective analysis of root causes will be more likely to identify performance issues in outsourcing relationships. Models may then form useful input to an action plan to address the issues identified.

Reality Check: Some Common Process Improvement Myths

MYTH	REALITY
<p>The process must exactly reflect the model. Existing processes must be thrown away and replaced by model compliant processes. Process models require bureaucratic documentation. Process models expect all types of work to use the same process. Model based process development reduces budget available for projects. Model based process development means re-training all staff. Model based process development takes a long time to return the investment. Model based process development is a “one-off” task. Models embody ‘best practice’ Models require a ‘big design up front’ approach</p>	<p>Models suggest goals and expectations which can be met in many different ways. Models just provide a means of comparing processes with experience-based good practice. Models expect certain key things to be documented, but allow a low level of formality where appropriate. Models expect a set of processes to be available, covering the range of work types in the organisation. Successful process improvement reduces waste, making more budget available for development. Since the model is NOT the process, the skills and training needed for most staff will be essentially unchanged. The time taken to return the investment depends on improvement prioritisation choices, not the model. Improvement is always possible. Continuous improvement delivers benefit continuously.</p>

Failure to realise benefit from models and standards typically occurs not because of the model itself but because of poor management of knowledge and people. Good ideas and good practices may not be consistently applied across the organisation, contributing to a corporate inability to maximise the potential of its people. This can result in loss of skills and capability within the organisation, particularly a loss of innovative thinkers and value-creators, which compounds the problem.

The most common reasons for the failure to deliver benefits from improvement based on the adoption of quality models and standards include:

- lack of clear business objectives
- failure to engage effectively the creative, customer-facing and operations staff
- lack of appropriate skills and resources - and failure to develop capability
- lack of leadership and senior management engagement
- inconsistent decision-making, direction and communication
- inability to integrate and align multiple initiatives

Models and Standards: Model-Maker's Purpose

Understanding the model-maker's purpose is a first step towards intelligent application of the framework. The following section describes a selection of commonly used models. It is compiled from a UK perspective, and does not claim to be a comprehensive list of all the models and standards in use.

CMMI[®] Capability Maturity Model Integration

Web site: <http://www.sei.cmu.edu/cmmi/tools/index.cfm>

The Software Engineering Institute's Capability Maturity Model Integration (CMMI) has become established as a leading process performance model, particularly in software development. CMMI-DEV belongs to a suite of CMMI models which have many similarities and complement each other. See the Software Engineering Institute (SEI) website for details of what each model covers and the associated training and appraisal programmes.

Adminitrack for Issue and Defect Tracking - Click on ad to reach advertiser web site



Issue & Defect Tracking
Most Effective Solution for Professional Teams

“Got an eye on all of your project team's issues?”

Free 30-Day Trial Now Available at
www.AdminiTrack.com

CMMI for Acquisition

Designed for clients of software service suppliers that want to improve their ability to select and manage outsourced IT suppliers.

CMMI for Development

Designed for development organizations that want to improve their ability to develop software-intensive products and services.

CMMI for Services

The most recently developed CMMI model is designed to support a better organisational ability to establish, manage, and deliver ICT services.

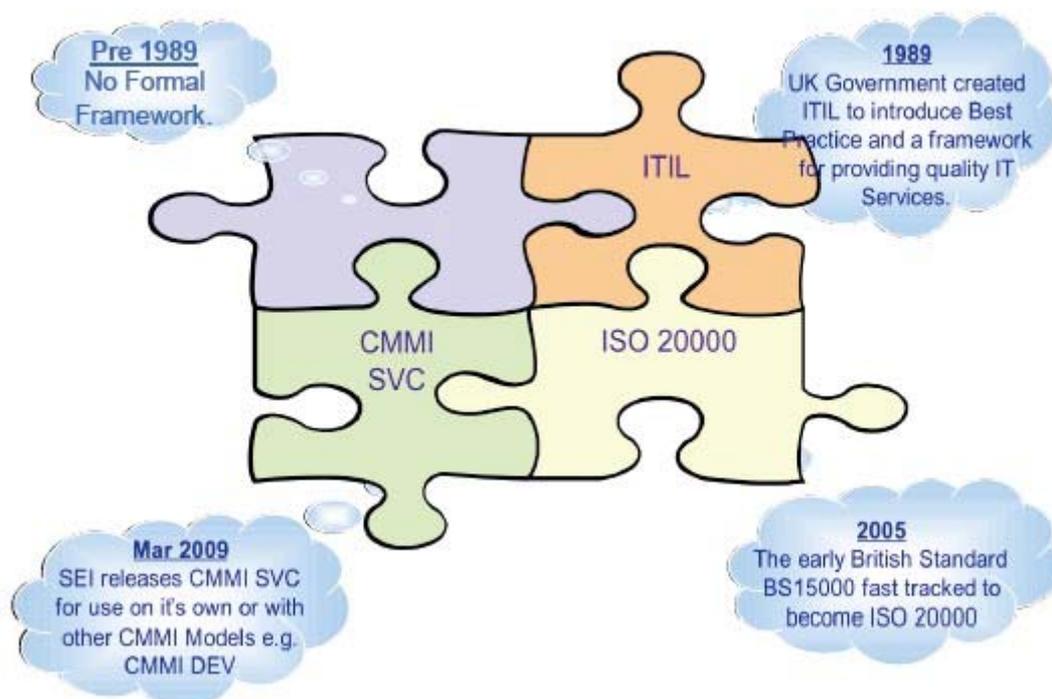
People CMMI

The People CMMI model was also developed by the SEI, although it does not form part of the CMMI Product Suite. The People CMMI is a maturity framework that describes the key elements of managing and developing the workforce of an organization. It is designed to evaluate progress from an ad hoc approach to managing people, to a mature, disciplined development of the knowledge, skills, and motivation of the workforce. The People CMMI appraisal method can be used alone or integrated with an existing process appraisal method.

COBIT Control Objectives for Information and Related Technology

Web site: <http://www.isaca.org>

COBIT is a framework for IT governance, control and risk management. The supporting toolset seeks to bridge the gap between control requirements, technical issues and business risks. COBIT emphasizes regulatory compliance, but it is not an audit methodology itself. It is used by organisations to reduce waste and improve assurance of value in ICT investment.



ISO Standards

Web site <http://www.iso.org>

ISO 20000

ISO 20000 is the global standard for IT service management. The ISO/IEC 20000 series helps service providers - large or small - to understand how to enhance the quality of service delivered to their customers, both internal and external.

The ISO/IEC 20000 series separates process performance from organizational form, size, names or structures, allowing good practices to be shared across market sectors.

ISO/IEC 27000:2009 Information Security Management Systems

ISO/IEC 27000:2009 provides an overview of information security management systems (ISMS) family of standards, and defines related terms. As a result of implementing ISO/IEC 27000:2009, all types of organization (e.g. commercial enterprises, government agencies and non-profit organizations) are expected to obtain:

1. an overview of the ISMS family of standards;
2. an introduction to information security management systems (ISMS);
3. a brief description of the Plan-Do-Check-Act (PDCA) process; and
4. an understanding of terms and definitions in use throughout the ISMS family of standards.

ISO 9000 Quality Management System

The BS EN ISO 9000 family of standards describes the fundamentals of quality management systems (QMS) and defines related terms. It sets standards of quality management regarding all the features of a product (or service) which are required by the customer.

This international standard is applicable to the following:

1. organizations seeking advantage through the implementation of a quality management system
2. organizations seeking confidence from their suppliers that their product requirements will be satisfied
3. users of the products
4. those concerned with a mutual understanding of the terminology used in quality management (e.g. suppliers, customers, regulators)
5. those internal or external to the organization who assess the quality management system or audit it for conformity to the requirements of ISO 9001 (e.g. auditors, regulators, certification/registration bodies)
6. those internal or external to the organization who give advice or training on the quality management system appropriate to that organization
7. developers of related standards.

ISO/IEC 12207:2008 Software Life Cycle Processes

ISO/IEC 12207:2008 establishes a common framework for software lifecycle processes whether these are performed internally or externally to an organisation. It contains processes, activities, and tasks that are to be applied:

1. during the acquisition of a software product or service
2. during the supply, development, operation, maintenance and disposal of software products. Software includes the software portion of firmware.
3. to the acquisition of systems and software products and services
4. to the supply, development, operation, maintenance, and disposal of software products and the software portion of a system. Those aspects of system definition needed to provide the context for software products and services are included.

ISO/IEC 15504 SPICE Software Process Improvement and Capability Evaluation

ISO/IEC 15504 also known as SPICE is a set of technical standards documents for the computer software development process and related business management functions. ISO/IEC 15504 initially was derived from process lifecycle standard ISO 12207 and from maturity models like Bootstrap, Trillium and the Capability Maturity Model (CMM[®]).

The first versions of the standard focused exclusively on software development processes. This was expanded to cover all related processes in a software business, for example, project management, configuration management, quality assurance, and so on. The list of processes covered, grew to cover six business areas:

1. Organizational
2. Management
3. Engineering
4. Acquisition supply
5. Support
6. Operations

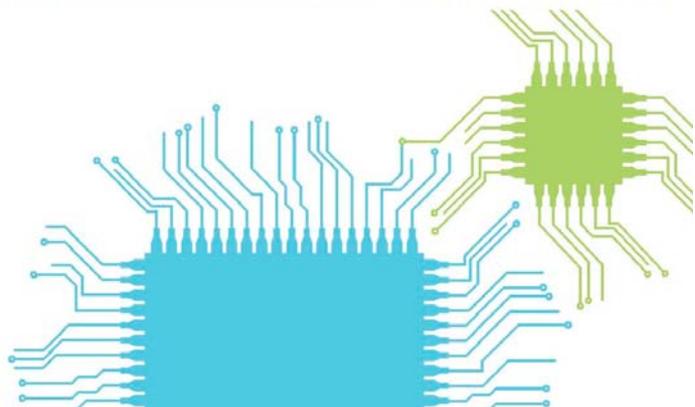
Conquer the complexity of software engineering with PTC Integrity - Click on ad to reach advertiser web site

PTC[®]

Conquer the complexity of modern software engineering

Only Integrity, a PTC product, connects modeling and simulation to the engineering lifecycle.

Integrity
A PTC Product



In a major revision to the draft standard in 2004, the integral process reference model was removed and it is now related to the ISO 12207 (Software Lifecycle Processes). The issued standard now specifies the measurement framework and can use different process reference models. There are five general and industry models in use.

Automotive SPICE

The Automotive Special Interest Group, a joint special interest group of The SPICE User Group and The Procurement Forum, created the Automotive SPICE® initiative together with the major motor vehicle manufacturers. The intention was to develop a common framework for the assessment of suppliers in the Automotive Industry. The model is described in the publication, 'Automotive SPICE® Process Assessment Model' and the associated Process Reference Model.

ITIL Information Technology Infrastructure Library

Web site: <http://www.itil-officialsite.com/>

ITIL provides a practical framework for identifying, planning, delivering and supporting IT services to the business. It has been adopted by thousands of organisations worldwide, such as NASA, the UK National Health Service (NHS), HSBC bank and Disney™.

ITIL is a framework; an organisation cannot be certified to ITIL. However, a comprehensive qualifications scheme has been developed, supported by a wide range of service providers including examination institutes, accredited training providers and consultancies, software and tool vendors. This scheme can help organisations ensure that employees have the relevant knowledge, skills and techniques to use ITIL effectively, and that the entire organisation is using a common language.

ITIL practices also underpin the foundations of ISO/IEC 20000 (previously BS15000), the International Service Management Standard for organisational certification and compliance. Organisations can therefore implement ITIL to achieve organisational certification with respect to ISO/IEC20000 (see above).

PRINCE2 PRojects IN Controlled Environments 2

Web site: <http://www.prince2.com/>

PRINCE2 is a structured project management method developed by the UK Government Central Computer and Telecommunications Agency (since renamed Office of Government Commerce). It has become widely adopted both for IT and non-IT projects. There is a defined PRINCE training programme delivering certification at Foundation and Practitioner level. Organisations frequently require individuals fulfilling a project management role to have PRINCE certification.

Correctly applied, PRINCE can be an invaluable guide to good project management practice. However, its focus on a product-based planning approach makes it a blunt instrument which must be used with care to avoid encouraging *inefficient* and *ineffective* behaviour.

Scrum

Web site: www.scrumalliance.org/

Scrum is a framework developed to offer guidance on more flexible and responsive ways of managing workflow. It supports cross-functional, self-organising teams who have no use for bureaucratic project management processes. In small, co-located teams, Scrum has a proven track record of success. Building on this success to improve workflow management in a wider context is work-in-progress.

Scrum contains sets of practices and predefined roles, the key ones being: Scrum-master; Product Owner; and Team Member. There is a defined training programme for the Scrum-master role, but it is frequently the case that the crucial role of Product Owner is given less weight and authority. The Product Owner is responsible for representing the interests of the business stakeholders. These stakeholders provide the ‘pull’ which keeps the Scrum team focused on delivering customer value. If this pull is absent, Scrum will tend to encourage local optimisation at the expense of end-to-end effectiveness.

Six Sigma

It is debatable whether Six Sigma should really be classed as a ‘model’ or ‘standard’, but its use is widespread and it is therefore included for completeness. The Wikipedia entry for Six Sigma reads: ‘Six Sigma seeks to improve the quality of process outputs by identifying and removing the causes of defects (errors) and minimizing variability in manufacturing and business processes. It uses a set of quality management methods, including statistical methods, and creates a special infrastructure of people within the organization (‘Black Belts’, ‘Green Belts’, etc.) who are experts in these methods.’

Six Sigma is frequently coupled with Lean, to indicate a focus on quality and cost reduction through removal of waste. However, the underlying principles of Lean customer focus and designed-in quality should remain paramount to drive through the benefits of a true Lean transformation. Too narrow a focus results in local optimisation, which can have a negative impact on end-to-end effectiveness

The Six Sigma business management strategy was originally developed by Motorola, USA in 1986. As it is derived from manufacturing, where the ability to reproduce items without variation is critical, its applicability to software-intensive systems is questioned by advocates of the integrated Lean-Agile value stream. Software is increasingly recognised as a business service, and the ability to respond rapidly to highly varied and changing user requirements, to innovate, to add or change functionality, is an essential attribute for many software developers.

TMMI[®] The Test Maturity Model Integration

Web site: <http://www.tmmifoundation.org/>

Because software development has traditionally sought to ‘test in’ quality rather than ‘design in’ quality, testing has become a significant part of the process. The Test Maturity Model Integration has been developed to provide a process maturity standard against which testing processes and practices can be measured. It was specifically designed to complement the existing CMMI framework, and assessment includes compliance with the relevant ISO Standard 15504. The TMMi is managed by an independent TMMi Foundation

TickIT

Web site: <http://www.tickitplus.org/>

The TickIT scheme has existed since the early 1990s to provide guidance on the interpretation of ISO 9001 quality management. **TickITplus** was launched in 2011 by the British Standards Institute's Joint TickIT Industry Steering Committee in response to the changes in today's world of software development. It is intended to offer a flexible, multi-level approach to IT quality and certification assessment and can be applied at whatever level is deemed appropriate to the maturity of the organization and the needs of its customers. If multiple IT standards need to be addressed, these can be covered under one certification arrangement.

© SMS 2011.

Acknowledgements to:

Grant Rule, SMS Exemplar Group

Roger Gamage, CPIS Ltd. <http://www.cpisltd.co.uk>

Jill Pritchett, JFP Consulting

SEI <http://www.sei.cmu.edu>

International Standards Organisation (ISO) <http://www.iso.org/>

Wikipedia

The Psychology of UX

Vanessa Carey, Caplin Systems, <http://www.caplin.com/>

Ever since I found the blog ‘What Makes Them Click’ [1] by Susan Weinschenk, I’ve been fascinated with her writing. I’m a natural analyst, much to some people’s dismay, as I mentally poke and prod people till I really understand what drives people to behave the way they do. This has led me to study Product Design, Anthropology and to now be employed in the UX industry in an attempt to understand people and better their human experience. So when I read Susan’s post on ‘The Psychologist’s View of UX Design’ [2], I was fully engrossed in what she had to say on the matter.

Her article broke down several areas of study relating to the brain, memory and the visual systems in humans to explain how these are relevant to UX. I will break down Susan’s post and further explain my understanding of each of her ten points and what this has meant to my experience of UX thus far or the direction I’d like to take my own UX practices within a professional environment.

10 Things to About Human Psychology That Should Inform UX Design

1. People Don’t Want to Work or Think More Than They Have To
2. People Have Limitations
3. People Make Mistakes
4. Human Memory is Complicated
5. People are Social
6. People are Easily Distracted
7. People Crave Information
8. Most Mental Processing is Unconscious
9. People Create Mental Models
10. People Understand Visual Systems

1. People don’t want to think or work more than they have to

Firstly, I don’t think many people are going to debate this point if they are really honest with themselves. I’m not arguing that we can’t all have bursts of motivation and be hard-workers, but *ultimately* we are all lazy (or ‘efficient’ as Susan puts it).

And there’s a very good reason for that. Throughout our evolution, we’ve managed to survive longer if we successfully conserved our energy. This means that we would focus on exerting the least amount of energy possible to attain our core needs (water, food, sex, shelter and protection against danger/threat) as we couldn’t be certain where our next meal was coming from and if we’d have a calorie shortage.

Today that’s clearly not the situation with food in every corner shop, but the behaviour is still ingrained in our genes and we can’t shake it off easily. If we take the example of hard-workers who may contest that they are not lazy, the fundamental reason for their effort to climb up the career ladder is money which equates to being able to purchase our core needs (water, food, sex,

shelter and protection against danger), so perhaps being a career-person is the laziest choice for attaining those essentials (rather than say building a house for yourself, digging a well to get water or hunting a caribou for hours).

So now that we've established the idea that people are all ultimately lazy...

Let's explore some examples of how people are lazy on the world wide web

Susan references Steve Krug's book, *Don't Make Me Think* [3], where he states that 'people typically glance at websites and scan the content, clicking whatever they first see that catches their interest or vaguely resembles what they are looking for, rather than reading the whole page.'

Absolutely. You know it! I'm sure you've been to a badly designed website before where you get to it, there is a plethora of text bombarding your eyes, you get overwhelmed and quickly scan for a word that relates to what you are looking for and starting clicking away. A few clicks forward, a few clicks back and you are horribly lost. You leave the website. This approach happens with not-so-badly designed websites as well though hopefully you will find the content you are looking for on these in a short amount of time.

People like short cuts (a faster way of doing something). Particularly if they have to do the task over and over. This is just another way to conserve energy through using your mind to find a lazy-friendly approach. However, interestingly, Susan points out that if it takes too long or too much effort for you to discover the short cut, people will default to the tedious, lengthy approach. In other words, we can be lazy about finding a solution that will allow us to be even lazier. Ha!

A few web examples of short cuts might be when you are filling in a form, you might copy and paste data (such as your email), or have auto-complete fill it in for you. Similarly rather than remembering all of your passwords you may use a password-storage system in your browser that also auto-saves your passwords for you. However if initiating these services is too confusing to find, you may just take the extra time to type out your email address twice in a row or rattle off 5 passwords before you find the right one.

In the same vein, the term '**satisficing**' was coined by Herbert Simon [4] that combines the words 'satisfy' and 'suffice'. The formal definition for 'satisficing' is a decision-making strategy that attempts to meet criteria for adequacy, rather than to identify an optimal solution. As we do not possess the cognitive ability to weigh all the options, **we settle for what is good enough rather than waste the energy trying to identify the best solution.**

A web example of this could be when you are searching on Google — rather than go through the extensive pagination to find the *perfect* answer to your query, you are likely to settle for some item on the first three or four pages.

So those are a few examples of people's laziness factor, but...

How can we consider this when designing the UX of websites?

Simple: allow people to do as little as they have to (including thought) to achieve the desired outcome on your website.

For instance you could:

- Make your website more scannable by using **less text and adding more imagery** (as humans are 90% visual creatures).
- Bouncing off the point above, **show examples** or explanations through imagery rather than text.
- Create **clear and visible** navigation with wording that is familiar (i.e., don't be overly imaginative with your wording as it can confuse people who are used to web standards like 'home' and 'contact'. But that also doesn't mean you should talk like you are a computer in crazy code "Error 3.106 is faulty because of PHP.zipfile corruption 27695..." You get the point :)
- **Provide defaults and short cuts** that cater to common work flows. For example, if you are creating a email website, make the buttons/access to the area where you 'compose' an email or 'read' emails dominant over other less important tasks.
- **Make things that are clickable, look clickable** [5] and things that aren't, not. For instance, make links have underlines or different states when hovered over.
- **Don't overcrowd your website** with content. Give people a little bit of information and then offer the opportunity to learn more through subsequent clicks.[6]
- And most importantly, find out your users' true needs so you don't overcrowd the website with unnecessary functions, content and clutter. Design for their desired needs and work flows only!

2. People Have Limitations

If you are multi-tasking while reading this article, please stop! I need your full attention. Seriously. Let's find out why.

Multi-tasking is a modern trend where we try to mentally manage two or more tasks simultaneously with the (erroneous) belief that we are optimising productivity. For instance, you might be at work checking your emails, while also filling in an excel spreadsheet, reading the news, chatting to your co-worker about internal changes, drinking a cup of tea, and browsing some websites to gather research for a presentation you are putting together– all at the same time. This might be a typical work day for most people.

At home, media saturates the multi-tasking experience even more with people often watching TV while browsing numerous websites on their laptop and texting a friend on their phone. This kind of behaviour has become so commonplace in our society today that most people would agree they do this, and **most would also defend their ability to manage multiple things going on at the same time. But they'd be wrong.**

Don't believe me? Try this experiment. Think about the taste of chocolate while you mentally add 38 and 272. Mmm, 310 is my new favourite flavour! No, but in all seriousness, you couldn't do it. It's impossible.

The origin

...of the word itself helps us understand how far removed this method of operating is from the way the human mind works. In fact, the term 'multi-tasking' came about from the computing industry back in the 1960's to refer to the ability of a microprocessor to process several tasks at the same time. **It wasn't until 1998 (only 13 years ago) that humans started adapting the word to themselves.** So in retrospect it is a very new concept and an even newer practice for human beings to be undertaking.

But what about historically speaking? **Did our ancestors multi-task** without knowing they were “multi-tasking”? Sure. Human beings have always had the capacity to handle several things at once since the time of hunter-gatherers. Mothers would pick berries while feeding their infants, or preparing food while keeping an eye on their children. Men would have had complex, long hunts where they had to have mental maps of where their buddies were in hiding to optimise their approach in hunting the animal, simultaneously remaining quiet, exploring the land with their feet to ensure stability and preparing to attack. Because of this need, we developed part of our brain known casually as ‘the executive system’ and officially as **‘the prefrontal cortex’**.

This front lobe of your brain **conducts your focus by helping you ignore distractions and switch from task to task**. But nowadays it seems our assuredness in our ability has surpassed the ability itself. You might have experienced this if you ever have tried to type a message to a friend on a computer while someone else talks to you in person. Your mind tunes out the voice and while you can pretend you are paying attention, you will most likely have to ask them to repeat themselves.

The way the prefrontal lobe works is that **it can only process one thing at a time, but it can switch between two tasks very rapidly**. You can do more than one thing seemingly at the same time, but in reality you are ordering them and deciding which to do at that specific time. However the negative side to switching from task back and forth again and again is that it takes about one minute to recover our train of thought, breaking our concentration and making us unfocused. For full concentration on one task to be re-established, it can take fifteen minutes! In this mode we are only capable of superficially scraping the surface. This in turn is more counter-productive than anything else.

“The general understanding people have of multi-tasking is a bit of a misnomer. I’ve never seen any examples of anyone who can do three or even two intelligent tasks simultaneously,” says neuropsychologist Prof Laws

Nowadays the speed and amount that we multi-task has exploded due to technology. **Software often requires us to think about multiple things at the same time**. For instance, if the predictive text feature on your mobile phone is correctly amending what you are writing as you write it, you are forced to pay attention to two very similar tasks. And if the tasks are too similar they compete for the same space in the brain, and you mentally can NOT focus on both at the same time. Yet if one task is something more automatic or highly practised like walking or breathing, we can do another conscious task simultaneously as we require very little processing to perform the first function.

So why should we care?

Multi-tasking is very popular these days. It hints at productivity. It seemingly allows us to split our attention (most likely from one or more things we **don’t** want to do.... and potentially another we do – like watching TV while you do your homework). This makes the work become less tedious. While this practice can be easy to fall into, we should avoid it as it will take us longer to accomplish any one task, not to mention the quality will most likely fail.

Even more importantly, the brain needs time to recover between switching between tasks to gather its thoughts. Without this time, the individual will be over stimulated and quickly become stressed out with all of the effort they are giving to multiple tasks. Too much multi-tasking can condition the brain to an overexcited state which makes it hard to focus even when you want to. This makes for unhappy, unproductive and exhausted workers. Do you really want that?

How can we overcome this?

Well, the honest answer is: **we can't**.

“With such complicated tasks [you] will never, ever be able to overcome the inherent limitations in the brain for processing information during multitasking. It just can't be, any more than the best of all humans will ever be able to run a one-minute mile,” says David E. Meyer, director of the Brain, Cognition and Action Laboratory at the University of Michigan

We should respect the way our mind works and work in a similar fashion. From a UX perspective, we can design software, websites and all digital interfaces to **minimise distraction and focus the user's attention onto one task at a time**. This means you can't have all the tools in the toolbox in front of you when you work. We can give our users more of a holding hand as they are guided through a clear workflow that helps them accomplish one task, and THEN another... and another. One at a time, considering what communication and information is really worthy of interrupting your precious concentration and when you should seek out new data.

At my company, we are really starting to consider ways that we can incorporate this approach of one task at a time into our designs so that our users won't be as overwhelmed with the amount of data available to them. The financial world seems to LOVE throwing every single piece of data onto every possible screen they can, which must overwhelm users of these systems immensely. While seemingly giving the user a feeling of power with all of this information at their fingertips, realistically they are only able to focus on one item at a time, flitting from one to another to another to another to another, trying to remember what they were thinking just a second ago. Are you exhausted yet? I am.

One way we can 'trick' our mental system though...

is to multi-task tasks that don't share attentional resources, i.e. use different sensory inputs. Like visual and auditory, which can work together without interfering at times. One really cool example we found of this was a chart that has supporting sounds to reinforce the direction of movement of the graph, which helps reinforce understanding. Check it out for yourselves! [7] Imagine the potential for tapping into our user's various senses to help them quickly understand what they are seeing, as well as helping them focus, and in turn be more productive.

Simplifying a user's workflow

...and the maximum things they can do at one time, teamed with tapping into our various sensory systems may be the key to creating interfaces that are designed with the user in mind: a human being, not a microprocessor. Mono-tasking should be the new multi-tasking. Spread the word.

I attempted to multi-task when writing this part of the article for the irony and it took me four days of occasionally writing a sentence before I got fed up. On the final day when I decided to mono-task (it'll catch on ;)) I managed to compile this in an hour or so. Proof, it works!

3. People Make Mistakes

What exactly is a mistake?

“..a decision or action, or lack thereof, that we fear we’ll come to regret. They usually cause some degree of pain, loss or struggle,” says Mel Schwartz from Psychology of Today. [8] For a more software-based explanation we might say a mistake is simply something that is wrong or that causes a problem with a user’s normal workflow.

Yes, mistakes. We all make them. We wish we hadn’t. One snooze button too many. Spending money on things we shouldn’t. Having that extra cocktail at work drinks and breaking out your embarrassing dance moves....I’m divulging too much of my personal life. **But what about mistakes in the digital world? What happens when a user makes a mistake on a computer?** These are things we will cover in today’s post.

‘Assume people will make mistakes. Anticipate what they will be and try to prevent them.’

Susan Weinschenk stated the above in her ‘Psychologist’s View of UX’ post, the inspiration for this article.

Why would we want to prevent them though?

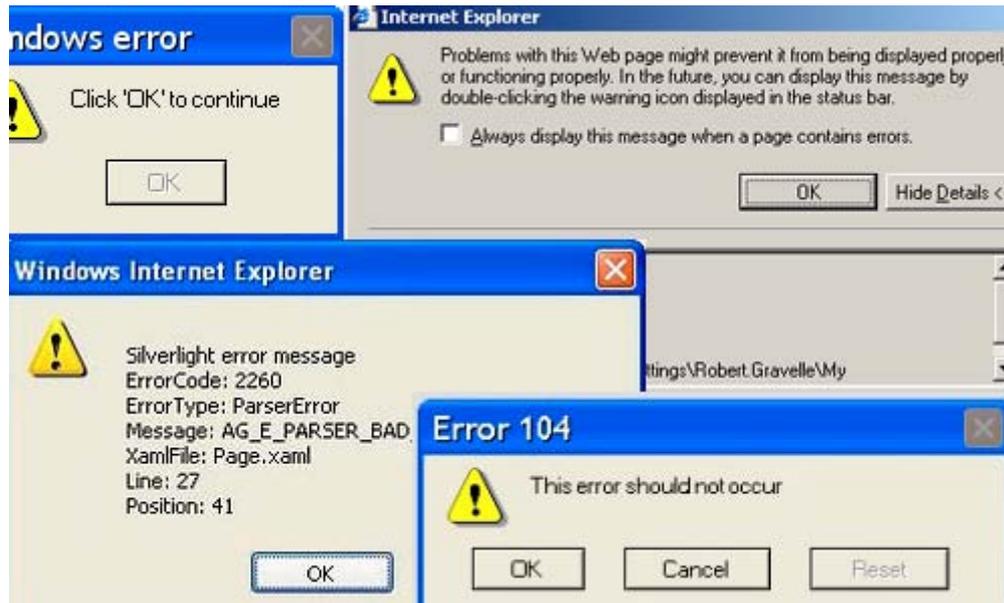
Well aside from mistakes causing users great frustration and pain when interacting with a system, **mistakes can be rather costly, particularly in the financial sector** where a user might be trading thousands, or even millions, of dollars with one click. It’s imperative when designing the user experience for financial interfaces to make the workflow easy-to-use and error-free.

So how do we prevent errors?

The best error message is no error message. What this means is that a system that is designed well will not allow the user to make an error to begin with. The main way we can accomplish this is by **predicting what mistakes a user might make**, based on knowledge gathered by researching the environment they operate in and their needs with the system. We can then adapt our designs accordingly to avoid allowing those mistakes to be made in the system.

If the task the user will be conducting is very complex or error-prone, a further approach is to **break up the task into smaller steps** so that each step can act as a quality gate before the user is allowed to move onto the next. We often see this design solution in online payment portals on retail websites, like Amazon.

How do we treat errors in the system?



Can you read that? Honestly? Even you developers out there (who often are the ones who have the mission of writing these error messages). You are human too - we don't talk like that. The first crucial step to dealing with errors in a system is **speak in human language!** Explain that **an error has occurred, what the error is, how the user can correct it and where they can go for more help to fix it.** In plain language. When something goes wrong in a system, it's of the highest importance that the user knows what to do about it.

The below image is great examples of clear, human language communicating to the user the problem with visual cues (symbols and colours) and sometimes even contextual hints as to where the problem went wrong (i.e. a highlighted password field when the password is wrong)



We should also **allow users to UNDO actions, such as mistakes they have just made. Ctrl-Z, anyone?** It's been a godsend for me. Users need autonomy within a system and this can only be achieved if the system is so well designed that the user can't get so lost down a path they can't find their way back to where they were. Allow them to undo and reverse steps. Similarly the 'ESC' key can be used to exit a current task that isn't yet completed, to prevent the damage from being done.

In reality, it is near impossible to create a completely error-free system that guarantees the users won't make mistakes. But why?

Because people make mistakes and UX Designers are just people.

"If Ernest Hemingway, James Mitchener, Neil Simon, Frank Lloyd Wright and Pablo Picasso could not get it right the first time, what makes you think that you will?" –Paul Heckel

Without sturdy user research, the designer will have a lack of knowledge of the user needs and this can result in an unusable design. That's why we as designers **need more time. More time for research with the users and more time for testing with the users.**

At a 'Lean UX' workshop I went to last weekend Janice Fraser, one of the founders of the famous Adaptive Path UX consultancy, said rather powerfully in regards to the Agile methodology,

"Don't throw the design out into the world and hope it works. I no longer want that responsibility. We are supposed to get it right the first time, yet the developers get to do it over and over again."

UX Designers need the allowance of time and budget to be able to **test our prototypes on real users before the designs are fully coded and completed for product release.** That's not to say that a design shouldn't be coded to test if it's implementable or to test the functioning prototype on a user, but we need time to test the designs before they are finalised.

That way we can see what errors the user might run into or what errors are prevalent in your design. And then we can **iterate and improve on our designs.**

Not enough time?

"The joy of an early release lasts but a moment. The frustration of an unusable system lasts forever."

Let's do it right, step by step, and try to design a system that is *mostly* error-free.

4. Human Memory is Complicated

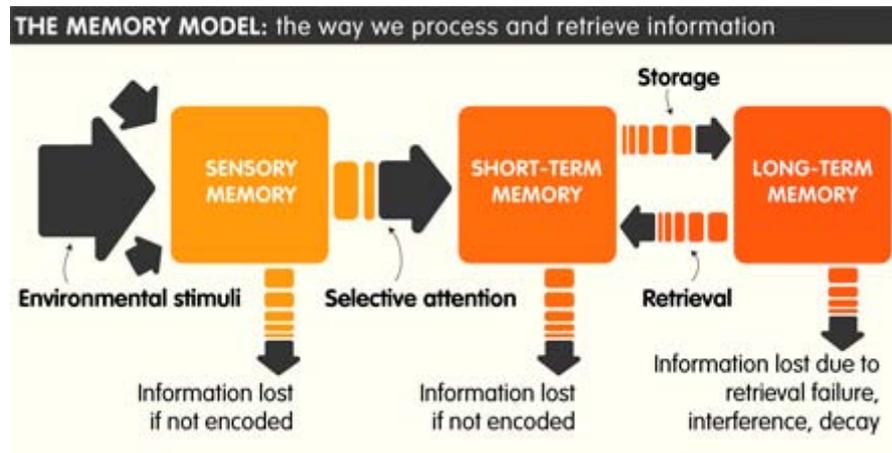
"Memory is deceptive because it is colored by today's events." - Albert Einstein

So what is memory?

Memory in its most basic definition is the ability **to encode, store, and retrieve information and experiences.** It is essential to our lives. We rely on memory to help us remember our identity and that of others, our past experiences, and potential threats.

How does it work?

There are three memory systems: **Sensory, Short Term, and Long Term.**



All of the time, our five senses are taking in a surplus of environmental stimuli, filtering it, and discarding irrelevant information. When the stimulus has ended and an impression remains, it is temporarily recorded in our minds. This is known as Sensory Memory. It often happens unconsciously and only lasts split seconds.

Our mind then goes through two processes to get the information from Sensory Memory to Short Term Memory. The first process is **pattern recognition**, where we actively search through our Long Term Memory to find a matching pattern for the new raw data. The second process involves **focusing our attention on the stimulus** until it moves into our **Short Term Memory** where it is encoded primarily acoustically and occasionally visually.

Our Short Term Memory typically **only lasts 30 seconds and has limited capacity** to store information because it all occurs in the frontal lobe of our brain. You remember the prefrontal lobe from my previous section, don't you?

There is no definite number to how many items we can store in our STM at once. One famous theory suggested there was a capacity of 7 plus or minus 2, but this has been disproved [9] and is now suggested to be even lower. By **'chunking' information into meaningful groups** (think telephone numbers remembered in groups of 3's or 4's) we can optimise the "space" in our STM.

After we have stored the information temporarily in our STM we can encode the information semantically by **creating mental associations and with frequent rehearsal** in our **Long Term Memory** which is spread all throughout the brain in our neural connections.

The final act our memory performs is **retrieval**, where we pull the memory out of storage and **reverse the process of encoding**. But this isn't always a straight forward process....

Distorted Memory

"The process of remembering involves the retrieval of information which has been unknowingly altered in order that it is compatible with pre-existing knowledge." - Neurophilosophy

Let's look at one study specific to Short Term Memory that demonstrates how our memory often fails us:

*"In a study conducted by Intons-Peterson et. al (3), both younger and older adults were asked to remember the following list: candy, sour, sugar, bitter, good, taste, tooth, nice, honey, soda, chocolate, heart, cake, eat, and pie. They were then asked to take a minute to write down all remembered words. The next test entailed that subjects consider the words taste, point, sweet and identify which word was included in the original list. **An overwhelming 80-90% of participants confidently, but incorrectly, selected the word sweet.** While the word sweet yields a close association to the presented collective of words, this association should not nullify the fact that its selection still results in a memory malfunction.Incidents such as these are frightening reminders of the **memory's fallibility.**"*

Memories change and adapt based on your current views

Yes, our memory can be deceptive and as Daniel L. Schacter, a Harvard psychologist, explains in his renowned book 'The Seven Sins of Memory' [10] it is deceptive in seven distinct ways:

1. Transience
2. Absent-mindedness
3. Blocking
4. Suggestibility
5. Bias
6. Persistence
7. Misattribution

But I'll only focus on the ones particularly relevant to the web:

Transience

The **weakening or loss of memory** over time, regardless of age. For instance, you might clearly remember what background you chose for your MySpace profile back in 2001, but now it is (fortunately) a vague memory. "I'm pretty sure it was pink and had dancing stars with anime kitten faces on them...."

Suggestibility

Information that is inaccurately added to memories due to leading questions and suggestion. This commonly happens to eyewitnesses to crimes being repeatedly interviewed as their stories change based on the questions, but can also happen during the interview stages of UX research with SMEs and users. "So would the best solution be something like an Apple product interface?"

Bias

The editing or **rewriting of past memories skewed by our current knowledge** and beliefs. "*I always knew online dating would become mainstream.*"

Misattribution

Assigning a memory to the wrong source or context. For instance, saying “I heard on the news the other night that Farmville was responsible for brain cell depletion in its regular users..” when actually you read it on Twitter.

But how does this affect UX design?

All of the above ‘sins’ of memory are particularly **relevant to the user research stage**. When one conducts interviews with SMEs or users, they need to **take what is said with a pinch of salt: “self-reporting” is often inconsistent** due to the fact that **memories are not as accurate as we’d like to think they are**. In addition, if the user or SME hasn’t used a similar system in a long time, their memory may have decayed. Memories we don’t use are erased from our Long Term Memory.

In another study by Sir Frederick Bartlett, a 20th century British psychologist, he asked subjects to read a Native American folktale and re-tell the story several times throughout the course of a year.

What he discovered was that the subjects **reconstructed the tale to fit with their personal biases** and beliefs:

“Participants omitted information they regarded as irrelevant, changed the emphasis to points they considered to be significant, and rationalized the parts that did not make sense, to make the story more comprehensible to themselves. In other words, memory is reconstructive rather than reproductive.” – Neurophilosophy

As we can see, **it’s almost to be expected that human memory will err**. This is why it is so, so, so crucial to **conduct observational user research** in place of (or in addition to) user interviews. In short, **watch what they do, not what they say!** Remember that ;)

During the design phase, we should also pay attention to the ‘Seven Sins of Memory’. With Short Term Memory only lasting 30 seconds at best, a conscientious interface would **ensure that the users will not have to remember every step in a task flow**, but will be guided easily through it.

In addition, the limited capacity for STM implies we should **not bombard users with a surplus of information all at once** that they will never be able to focus on and remember for more than a few seconds. Designs should **direct the users’ attention to the task at hand**.

With visual design tools like **distinctiveness** (making information stand out), **primacy** (important information first), **frequency** (information repeated as needed), and **associations** (positioning information/objects to suggest relationships- think ‘chunking’) we can help our users store more information in their Sensory and Short Term Memory. This will ultimately help them more **effectively navigate our system and pinpoint key information they need to remember**.

One final thought on an approach to making the system more user-friendly: **maintaining consistency** throughout navigation menus and with interaction patterns means that **the user only has to program their mind once** to this behaviour.

“If we remembered everything, we would have too much information to sift through to find the important things that affect our livelihood.”

So now that you’ve finished reading this section, read it again. Over and over, until your mind can recite its key points and hopefully then it will be encoded for Long Term Memory. ;)

5. People are Social

Humans beings are social animals.

We are fundamentally driven by the need to belong and to have the approval of our peers. This urge to connect is at our core because of its ability to raise our chances of survival. When we act in accordance with the beliefs, suggestions and commands of the collective, it helps us to reach our goals, including the most primal of sustenance and shelter. Since the nomadic times, when we began hunting in our immediate families we quickly learned that joining forces and hunting together in larger groups meant bigger kills and greater chances of avoiding hunger.

“Humans are social animals and the urge to connect is basic survival, practically, emotionally, and genetically.” - Pamela Rutledge, Ph.D.

For over 100,000 years, we have traded and exchanged between groups in order to draw upon other’s specialisation and raise each other’s living standards.

We all know little bits of information, but none of us know everything.

Through exchange, we’ve surpassed our own knowledge, **creating the ability for us to do things that we (individually) can’t even comprehend**. For instance, we all know what a toaster is.. but how many of us can fashion one entirely on our own? We’d need to know how to drill for oil, how to make plastic, how to wire electrics, how to create heating elements, how to create screws, how to extract and melt metal, and the list goes on.

In order for humanity to evolve, it’s **not important how intelligent the individual is, but how well we communicate and cooperate** as a people. By evolving to communicate and have language, we became even more connected and increased our chances of survival.

We could warn each other of danger, guide each other and share wisdom. But with this ability to speak, also came the need to be heard.

Humans want to be heard

...in order to share emotions and ideas, and to have these emotions and ideas validated. We look for guidance from others on what we should do and how we should act. This has always been and will always be the case.

“It’s a basic element of humanity to want to be heard. Communication has evolved to where it is today because people fundamentally want to communicate. And not just communicate for its own sake, but to be heard and validated. If that weren’t the case we wouldn’t have Twitter and Facebook.” -Brad Waters, Psychology Today

Another way that we learned to cooperate was through **imitation**. When we observe someone doing an activity, our brain has ‘mirror neurons’ that **mimic the activity as if we were doing it**

ourselves. Through this imitation we quickly learn new skills and behaviours, from birth all the way through adulthood.

Humans are undoubtedly social beings and because of **this we will always use technology to be social** as a form of self-expression. If we look at the history of the internet, we can see how this happened as the internet itself was intended for military purposes but evolved to connect the world socially. Since its birth, this happened over and over again, from IRC to BBS systems to SixDegrees and Friendster and Myspace; to Geocities, LinkedIn, StumbleUpon, Flickr and Facebook. We have adapted the internet to speak our language; a language of interconnectedness and sharing.

Today, nearly **four out of five web users visits a social networking site** on a monthly basis. Twitter estimates that it has at least **325 million users every 24 hours**. Facebook claims that its users spend over **700 billion minutes** on the site each month.

We communicate online. We share online. And **we look for guidance** online. With regards to the last one, we can see this brilliantly demonstrated with something called ‘The Amazon Effect’, or ‘people-powered product research’. ‘**The Amazon Effect**’ is a pattern where internet shoppers commonly go to Amazon.com first, often skipping the content written by Amazon to scroll down to the user reviews.

“I already know what it’s going to say, it’s going to tell me how great their product is. Why would I need to read that? If I want to know the truth, I have to read what other people like me thought about it.” - ‘Designing for the Social Web’, Josh Porter

Users look at other people’s experiences with the product and find guidance on whether or not to invest in it themselves, thus **learning from our shared experiences and knowledge**.

But how does all of this affect UX?

From the initial stages of the UX research, we can see the importance of **user observation based on our skill of mimicry**. By observing a user in person, our brains imitate their actions allowing us to better comprehend the activities they partake in. This strengthens our understanding of their needs and task flows, and **allows us to create solutions with greater empathy** and clarity of the problem.

When it comes to designing the UX, we need to take into consideration the **necessity for a social outlet** within our website or application. Allow for greater social **interconnectedness** in your designs so that people can go to each other for **guidance** and advice within your application, such as with ratings, reviews, news and forums. Allow users to **forge helpful relationships**, be it with similar users or with customer support. Give people an awareness of the size of the **community** they operate in to give them a sense of **belonging** as well as the choice of where they want to fit in within the community by establishing their **profile**.

Social Nature Brings Innovation

Thanks to the social nature of humans and all of the connections people have been making through the use of technology, **we have invented and evolved more tools in the past 100 years, than we did in a million years back in our hunter-gatherer days** with the design evolution of our hand tools.

Today **everyone is able to have their ideas and allow them to be shared** on a global scale. It is because of this that we as a people are **accelerating our rate of innovation**, and we should encourage this in every way we can.

...So don't feel too bad if your boss catches you on Facebook at work. Tell him/her you are satisfying a core human need.

The second part of this article will be published in the Winter 2011 issue of Methods & Tools.

References

1. What Makes Them Click - <http://www.whatmakesthemclick.net/>
2. The Psychologist's View of UX Design - <http://uxmag.com/design/the-psychologists-view-of-ux-design>
3. Don't Make Me Think: A Common Sense Approach to Web Usability, Steve Krug, New Riders Press, 978-0321344755
4. Herbert Simon - http://en.wikipedia.org/wiki/Herbert_Simon
5. clickable - <http://en.wikipedia.org/wiki/Clickable>
6. New York Yankees website is a cluttered mess - <http://www.doobybrain.com/2009/11/05/new-york-yankees-website-is-a-cluttered-mess/>
7. Listen to the pattern - <http://www.bella-consults.com/sound-pattern>
8. Psychology of Today - <http://www.psychologytoday.com/>
9. The 7±2 Urban Legend - <http://www.knosof.co.uk/cbook/misart.pdf>
10. The seven sins of memory - <http://www.apa.org/monitor/oct03/sins.aspx>

HTML5 for Rich Web Enterprise Applications

Itzik Spitz, Visual WebGui developer, VP R&D Gizmox, <http://www.visualwebgui.com/>

There seems to be a consensus of opinion that HTML5 will be the next generation web platform to be embraced by all the major players, including Microsoft. Recent announcements indicate that Microsoft is on the route to de-emphasize its Silverlight solution in favor of HTML5. This is reinforced by a recent Windows 8 preview, which conveyed a very clear message as to the direction that Microsoft is taking.

As a result, HTML5 seems to have captured the attention of a sizeable proportion of the development community, including enterprise development departments. We are witnessing more and more organizations adopting an HTML5 strategy with their development decisions, and what's really amazing is that not one single browser yet supports HTML5 fully. Current overall browser support for HTML5, as reported by caniuse.com in June 2011 puts Chrome 12.0 in the lead with 89% compatibility, followed by Firefox 5.0 (84%), Opera 11.1 (74%), Safari 5.0 (73%) and IE 8.0 at the bottom with an overall supported function set of just 26%. IE 9.0 narrows the gap with a compatibility of 57%.

The mobile browsers are a little behind but not by far, with Opera 11.0 Mobile and Android 3.0 already on 69%.

So why are business application developers and enterprises taking HTML5 so seriously? And what tools are available to implement data-intensive HTML5 applications? All you have to do is look on any web dev magazine or development tools site and HTML5 will be right there in the spotlight.

What's So Good About HTML5?

HTML5 addresses several major issues of interest to business application developers, the most outstanding being:

- We're all using "finger touch" input on mobiles and tablets in our free time with more devices becoming available every month; we are getting to like its convenience, and history proves that once we get to like something out of work hours, we want to use it at work as well. The "finger touch" experience requires HTML5 and JavaScript.
- Cross platform unity. An area of contention from the beginning of the web, but now bigger than ever due to the far more scattered and competitive browser and operating system landscape that includes mobile and tablet formats. This is illustrated in the approach taken by a growing number of enterprises that are encouraging workers to use their own mobiles and tablets in the workplace.
- Components that provide a 'native' way (i.e. naturally built into HTML) to achieve a RIA (Rich Internet Application) with less dependency on JavaScript and being completely independent of foreign components like plug-ins.

Yet, despite its enhancements, HTML5 is still just a markup language with stronger user interface features. It needs a developer to integrate it into the business application.

Having adopted the strategic goal of incorporating HTML5's new capabilities, the question that arises is how to do it in the most efficient and cost-effective way possible.

The Challenges Facing Developers in Implementing Rich Web Enterprise Applications

The benefits of a browser based front end for a data-centric, cloud-based, AJAX-rich application as opposed to other information system architectures needn't be mentioned here. It's sufficient to recognize that, that's the trend and it's not going away; in fact it's getting stronger, with all major vendors lining up to support it and rightly so. But it is worth revisiting the challenges of web and web-based cloud applications; because they are the reason the different development frameworks look like they do. A brief summary follows:

1. Web apps are a multi-platform, multiple-protocol architecture

A web application is client/server architecture where the application and the interface reside in different worlds – different countries, different languages, different operating systems and different browsers – *many* different browser vendors and versions, connected by a communications channel. Yet they need to appear and behave as if they were one unified body.

This point is at the root of all others listed below.

2. Security

The security issue of AJAX rich web applications is really quite simple to describe, if somewhat less so to achieve. The application running on the server must control the application running on the browser while not allowing the browser application, or any other person, program or anything else to maliciously influence the application code or database using exposed client end points.

3. State

A close second to security in the developer's list of concerns is the issue of application state. Every business application must exercise dynamic changes in state because if nothing dynamically changes, there are no processes and without processes, you have no information system. These dynamic changes are the substance of business and they include things like the execution of credit card transactions, changing status of subscribers and updating prices. The problem is the synchronization of state between server application and browser, without which you would be able to make a transaction in the browser that the server would never know about.

4. Performance

Performance issues stem from the very nature of the distributed computing architecture that bridges different locations through a network, thus introducing latency. They are amplified still further when trying to solve the security and state challenges.

5. Cross Platform Compatibility

One application must serve clients running on at least 4 major operating systems, tens of browser types and versions, numerous display configurations and new devices every day.

6. RIA

Rich Internet Application. That means user-friendly controls, fast response time to make the GUI work for the AJAX rich web, like it did in the past for the desktop.

HTML5 to the Rescue and a New Challenge of Implementing It

HTML5 doesn't address all of the AJAX rich web developer's woes. However, it does offer performance benefits by using native features, a RIA user experience and, above all, compatibility over multiple browsers.

Instead of creating a new proprietary standard, HTML5 presents a native extension to classic web browser development. It adds native elements like date picker and slide bar and some very powerful capabilities such as canvas, animations, vector graphics and plug-in-free media playing. In addition, it complements some major capabilities lacking in today's browsers such as the ability to maintain constant communication with the server (TCP style) and persistent client-side storage.

That is reason enough to develop HTML5 web pages, but since the announcement in June 2011 by Microsoft that the upcoming Windows 8 (slated for release at the end of 2012) will feature a developer platform based on HTML5 and JavaScript, that makes HTML5 look that bit more worth taking note of.

And it's not just talk. IE 9.0, which incorporates HTML5 to more than twice the degree of IE 8.0, builds on top of the Direct2D 2D graphics API and the result is fast running HTML5 on IE 9.0.

While HTML5 helps meet the needs of business application development, it also introduces additional challenges. These can be outlined as follows:

1. How well the development framework can be made to produce HTML5 code.
2. Use of available development personnel skills, the efficiency of the IDE and the cost of the learning curve in terms of time and costs.
3. The handling of the screen size, layout and style.
4. Correctly putting together the JS and HTML5 code needed to enable finger touch app operation.

We will take a look at the available application frameworks and see how easily they integrate with HTML5.

Which of the existing frameworks is Best for HTML5 Data Centric Applications?

Without going into the intricacies of the different frameworks, we will look at the options available and the relative advantages of each approach from the point of view of HTML5 development.

Server-Based Frameworks

While not the only one around, Google's GWT is the most prominent server-based HTML5-compliant framework in use today. The coding is done in pure Java and this is processed to generate JavaScript which runs in the browser in an AJAX framework. The web page design is done separately by client-side programmers.

Advantages: Object-oriented development, good debugging tools at the Java stage, optimized for HTML5, reasonable data binding capabilities, plug-in free.

Disadvantages: Multiple programming skills required, security vulnerabilities, no separation of business logic from webpages, complex and error-prone data binding between server and client, debugging difficulties in run-time, cannot handle state.

Suitability: Good for non-LOB applications; security complexities with data centric LOB applications.

Client-Based Ajax Frameworks

These are frameworks based around writing code to run on the client. Development is pure JavaScript / HTML5. Database access and update is commonly done through stateless web-services or other types of stateless dynamic backend (i.e. active server pages).

Advantages: Developer-written HTML5 code for maximum HTML5 usability, native client debugging, handles state, plug-in free.

Disadvantages: Security vulnerabilities, no separation of business logic from webpages, complex and error-prone data binding between server and client, transmission of all variable page content due to the stateless nature of the server, thus increasing server load.

Suitability: Good for static websites and non LOB applications; insecure for data centric LOB applications.

Client-Based Plug-In Frameworks

There are 3 solutions of this kind – Microsoft's Silverlight, Adobe's Flash/Flex combination and the now largely defunct Java Applet. The application code is compiled to a runtime module and this is run in the browser by the appropriate installed runtime environment known as the plug-in. Database access and update is commonly done through stateless web-services or other types of stateless dynamic backend (i.e. active server pages).

Advantages: Object oriented development, (Java for Flex and applets, .Net for Silverlight), free from browser incompatibilities, good debugging capabilities, availability of Java and .NET developers, management of state on the client, capacity to implement permissions with relative ease, some degree of access provided to local devices.

Disadvantages: Dependency on the user to download and install the plug-in (and the fact that not all do), load time overhead, high bandwidth requirements due multiple threads of activity creating a run-time overhead, competes with pure HTML5, which is gaining increasing support, cannot handle server-side state, questionable IP and data security, performance limits when serving large numbers of users simultaneously, largely due to the stateless server.

Suitability: A useable, but far from perfect solution for data centric LOB applications. Excellent for non- data centric apps such as games, media, editors, local tools etc.

The Server-Driven, Thin Client Hybrid

When looked at in a broad perspective, neither the client-based or server-based solutions offer all of the following in a single package:

- Uncompromised security
- Automatic HTML5 code generation
- Cross-browser compatibility
- Maximum use of current enterprise developer skills (with .NET and Java expertise)
- Strong debugging capabilities

- A visualized component-oriented development process
- Visualized data binding at design time
- Efficient run time data interactions
- Maintaining high performance for a large number of users in an enterprise-sized organization

A new approach has recently appeared on the HTML5 landscape that takes a little from all the above to produce a comprehensive solution incorporating all of the above requirements. The solution is strictly server driven so as not to put the application or data in jeopardy by allowing distributed clients to have even a hint of a chance of interfering with the intended application flow or data content. The client layer is pure display without any hidden identifiers; every action taken or piece of data entered is expected in advance and controlled by the server application.

Up to now, the above definition sounds much like classic server driven dynamic HTML non-AJAX web pages. But add the purpose-built static client code – AJAX / HTML5 - that acts as an ultra-thin but highly flexible, desktop-like client, containing no identifiers, but rather 'serving the server'. The static nature of the code block is major factor in making the application secure at the browser side and facilitates debugging too.

Add to that a single communication channel to handle all browser-server requests for a session, so, the client speaks only to the application server and the server speaks to any other services required.

Add all that together and you receive a kind of server-client hybrid, something that resembles .NET, but on web architecture instead of client/server.

The server-driven, thin client hybrid approach has promoted a creative thought process among dev tool vendors and is generating interest from companies wishing to develop high-reach, data-centric business applications for enterprises.

There is a pretty varied selection of tools already out there answering the needs of those seeking to create or migrate to HTML5 and they have more promises in the pipeline for the months ahead. They differ from each other in character including their approach to virtualization, security, performance, maintainability and risk. Some are more suited to the creation of cool browser gadgets, others to data centric enterprise applications. One way or the other, these solutions ensure that no one gets left behind with the new web standards.

Conclusion

In this article we've discussed how HTML5 introduces markup-level functionality for rich graphics, animation and web multimedia and supports a richer functionality for web applications through:

- Native UI application controls like drag and drop and touch screen
- Independence of proprietary plug-ins
- Geo location tags
- Built-in support for multiple clients - desktop, mobile and tablet, and the ability to support exciting new formats that might be waiting round the corner

However whilst there are still limits to HTML5 such as standardization issues (like different browsers supporting different video formats) and its dependency on CSS and JavaScript (both having their own compatibility problems), people are becoming more mobile and require the ability to access programs across a wider range of devices. It's clear that HTML5 will be the next web platform to be embraced by all, including Microsoft.

HTML5 ensures we can move away from a world of plug-ins and enables us to create RIA's without the need for foreign components. This article has looked at HTML5 from the web programmer's point of view and briefly explored and analyzed some of the available development methodologies. Over the coming months we will be seeing the browser war heat-up and more and more devices being released with browsing capabilities. With all of the hype surrounding HTML5 the important thing to remember is that HTML5 is the start of the journey to make one code fit all, a journey that will benefit developers and end-users in the long run.

Manage Source Code Quality with Sonar - Click on ad to reach advertiser web site



Code has so much to say !



-  **More than 600 coding rules available**
Checkstyle, PMD, FindBugs
-  **Report Unit Test metrics**
Cobertura, Clover, Emma
-  **Project and Portfolio dashboards**
Drill down from portfolio to sources
-  **Replay the past and compare versions**

**Sonar is the
central place to manage
source code quality**



Visit the web site : <http://sonar.codehaus.org>
 Powered by SonarSource : <http://www.sonarsource.com>

Gradle

Evgeny Goldin, <http://evgeny-goldin.com/>

Gradle is a new and revolutionary build tool, based on the Groovy programming language. It is very different from existing tools like Ant and Maven in that it provides an extremely powerful capability to develop build *applications* using Groovy code and a compelling Groovy DSL. This allows to easily develop a non-standard build for any project, according to its needs, or to fall back to a more traditional convention-over-configuration approach also fully supported by the tool.

Web Site: <http://gradle.org/>

Version Tested: 1.0-milestone-3 on Windows 7 and Mac OS X 10.7.1, Java 1.6.0_26 x86/x64

License & Pricing: Open Source (Apache license), Free

Documentation:

- Web-site: <http://gradle.org/documentation.html>
- Wiki: <http://wiki.gradle.org/display/GRADLE/>, <http://evgeny-goldin.com/wiki/Gradle>
- “Building and Testing with Gradle”: <http://oreilly.com/catalog/0636920019909>

Support:

- Mailing list: <http://gradle.1045684.n5.nabble.com/gradle-user-f1431424.html>
- Issue tracker: <http://issues.gradle.org/>
- Gradleware: <http://gradleware.com/>

Installation

Install Gradle by downloading the corresponding distribution from <http://gradle.org/downloads.html>: “bin” if you only need to run Gradle and “all” if you wish to download its sources, documentation, and examples as well. After unpacking the archive, point GRADLE_HOME environment variable to the folder where distribution was unpacked and add “%GRADLE_HOME%/bin” (“\$GRADLE_HOME/bin”) to your system PATH.

Gradle is a multi-platform tool. It runs on every operating system on which Java can be installed. Make sure you have JAVA_HOME environment variable defined properly and you can execute the “java -version” command.

After adding Gradle’s “bin” folder to the PATH environment variable, you can now run the “gradle --version” command to make sure the setup process was successful:

```
#[02:07:45][~]$ gradle --version

-----
Gradle 1.0-milestone-3
-----

Gradle build time: Monday, 25 April 2011 5:40:11 PM EST
Groovy: 1.7.10
Ant: Apache Ant(TM) version 1.8.2 compiled on December 20 2010
Ivy: 2.2.0
JVM: 1.6.0_26 (Apple Inc. 20.1-b02-383)
OS: Mac OS X 10.7.1 x86_64

#[02:07:48][~]$ █
```

Additional installation instructions can be found at <http://gradle.org/installation.html>.

Groovy DSL

Gradle build file is normally called “build.gradle” and, as mentioned earlier, it contains Groovy code. If you're not familiar with the Groovy programming language, both the documentation available online at <http://groovy.codehaus.org/Documentation> and Manning's “Groovy in Action, second edition” (<http://www.manning.com/koenig2/>) are excellent resources for you to get started. However, usually you won't find much requirement for coding the build logic in Groovy since Gradle provides a great deal of sensible convention-over-configuration defaults and a powerful DSL (Domain-Specific Language) for this purpose. Following conventions, you only have to specify absolutely necessary data. The Gradle DSL allows you to use simple configuration structures, covered at <http://gradle.org/current/docs/dsl/index.html>.

Even though Groovy code can be used anywhere within Gradle, DSL is the preferable way of specifying build behavior. By using the reusable code blocks provided by Gradle authors, the DSL makes the build processes less error-prone and much easier to comprehend.

Sample “build.gradle” with Groovy DSL file is provided below:

```
apply plugin: 'groovy'
apply plugin: 'maven'
group          = 'com.test'
version        = '0.1'
groovyVersion  = '1.8.1'
springVersion  = '3.0.6.RELEASE'

repositories { mavenCentral() }

dependencies {
    groovy          "org.codehaus.groovy:groovy-all:$groovyVersion"
    compile        "org.springframework:spring-context:$springVersion",
                  'org.slf4j:slf4j-api:1.6.2'
    runtime        'org.slf4j:slf4j-log4j12:1.6.2'
    testCompile    "org.springframework:spring-test:$springVersion",
                  'org.spockframework:spock-core:0.5-groovy-1.8'
    testRuntime    'log4j:log4j:1.2.16'
}
```

This examples allows any basic Groovy project to be compiled, tested, packed and then installed to a local Maven repository by running “gradle build install” (Gradle-Maven integration is covered in the next section).

```
apply plugin: 'groovy'
apply plugin: 'maven'
```

loads two Gradle plugins: the plugin for compiling Groovy sources and the plugin for deploying project artifacts to a local or remote Maven repository.

```
groovyVersion = '1.8.1'
springVersion = '3.0.6.RELEASE'
```

specifies which Groovy and Spring versions, respectively, will be used. It helps to keep some of the third-party library versions outside of the actual code. As a result, it becomes much easier to update version numbers later on.

```
repositories { mavenCentral() }
```

defines that the Maven Central repository (<http://search.maven.org/>) should be used for retrieving all external dependencies required, as specified by the

```
dependencies {  
    ...  
}
```

section. It uses scopes that may be familiar to you from Maven: "compile" for compile-time dependencies and "runtime" for runtime ones. Unlike Maven, Gradle introduces new scopes: "testCompile" and "testRuntime" to split compile and runtime test dependencies as well. In fact, Gradle allows build developers and plugin authors to define any number of dependencies scopes, as demonstrated in the example above where new "groovy" scope is defined by the Groovy plugin.

Gradle Model

The Gradle model is what makes Gradle so powerful and flexible. All projects and tasks executed at run-time can be accessed within Gradle code through their APIs. Unlike other build systems where tool run-time "internals" are normally hidden from the build developer, Gradle exposes the build developer to everything pertaining to the build structure, running tasks and their dependencies, property values, and conventions used. In addition, Gradle doesn't enforce any specific ways of managing a software project or creating build artifacts; its defaults are similar to those of Maven, but while Maven makes it hard or impossible to step aside from any of its predefined rules, then Gradle makes it an easy task. Being able to customize all aspects of the build behavior, and to use the Groovy programming language together with a powerful DSL covering most of developer needs, makes Gradle a true one-of-a-kind build tool.

Maven, Ivy, and Artifactory integration

Gradle can integrate with Maven, Ivy, and Artifactory by performing the following actions:

- Retrieve third-party dependencies from remote Maven, Ivy, or Artifactory repositories.
- Deploy build artifacts to local Maven repositories.
- Deploy build artifacts to remote Maven, Ivy, or Artifactory repositories.
- Convert Maven POM files to Gradle builds with the [maven2gradle](#) script.

Summary

The Java ecosystem has developed rapidly and aggressively over the last decade. However, traditional build tools were not fast enough to realize how project complexities can grow and make any of the previous assumptions about build processes outdated and largely irrelevant. Gradle comes to the rescue at a very right moment in time when build processes involve much more than simply creating a packaged artifact. Today, companies deal with demands of *continuous delivery*, *continuous integration*, and *continuous testing* which in turn, may require high-performing and sophisticated build tools, capable of adapting to any business and technological need. If you wish to know more about Gradle, download my ["10 Cool Facts about Gradle"](#) presentation providing a more introductory material about Gradle, or visit a Wiki page at <http://evgeny-goldin.com/wiki/Gradle>, listing major Gradle features followed by links to relevant documentation and additional online resources.

Saros: An Eclipse Plug-in for Distributed Party Programming

Dr. Karl Beecher, Freie Universität Berlin, <http://www.mi.fu-berlin.de/w/Main/KarlBeecher>

Saros is a plug-in for the Eclipse IDE that enables two or more distributed programmers to develop projects in real-time over the Internet and share each other's changes. It allows users to communicate in a variety of ways as they collaborate and also to remain constantly aware of their partners' activities through various sorts of awareness information.

Web site: www.saros-project.org

Version tested: 11.7.1

System requirements: Java 1.6 or greater, Eclipse IDE

License & Pricing: GNU GPL. Available for free download at <http://sourceforge.net/projects/dpp> or via Eclipse update site <http://dpp.sourceforge.net/update>

Support: SourceForge-hosted mailing lists and trackers. The members of Software Engineering Workgroup at the Freie Universität Berlin (the authors of Saros) are also willing to negotiate support for organisations.

The Basics

There is increasing need these days for tools that allow software developers to collaborate across large distances. The rise of free/open source software, an increase in working from home, and organisations distributing their activities across the world are all phenomena which can pose problems for achieving close collaboration. For developers, the problems are centred around the absence of a number of things which are normally taken for granted when we work co-located. Distributed developers may be able to have discussions while having their own copy of the code before them, but as soon as one of them alters *their* code, they no longer have a common view. Even once you implement a method to propagate those changes conveniently in real-time, there are still issues. All those familiar cues you (sometimes unconsciously) feed each other – which we call “awareness information” – are missing: Which file is he looking at? Which line is she on? What parts of the file has she changed? Is he looking at the code or checking his email?

Saros provides a solution to these problems. As the product of a wider research programme into collaborative development, we created Saros as a tool to embody our findings in how best to achieve distributed collaborative development. The basic ability of Saros is to share one or more projects with some other developers by setting up a session. This simply means that one person (the inviter) asks the others to join them and the invitees each receive a copy of the project(s). Alternatively, if an invitee already has a copy, their version is synchronised so that it is identical to the inviter's copy. Once the session is established, Saros constantly passes messages between all the participants to maintain the session. What information is in these messages? What does a Saros session keep track of?

To help answer these questions, Figure 1 shows Saros in action and labels all of the major features and carriers of awareness information. The first four labels show parts that decorate the existing Eclipse components:

1. Files in the package explorer are marked by icons. They show which files are currently being accessed by users write-access: yellow icons means they are currently open, green means they are currently in focus. The projects which are shared are also marked.
2. Text that is currently **highlighted** by another user is marked in the editor. The colour of the annotation corresponds to the colour of that user, so you know who has done the highlighting. The current position of their cursor is also shared by marking it in your editor.

- Text that has been changed by another user is **also** marked, although in a slightly different shade of colour, so they can be told apart from highlights. Again, the colour of the marking corresponds to the user who changed it.
- The current view scope of other users is shown in the sidebars. From this, you can quickly determine which portion of the file a user can see right now.

Saros also provides its own Eclipse view to manage the session:

- This shows all users in the current session. Each entry has a context menu with further options. Furthermore, users who do not presently have the Eclipse application in focus are marked, so you remain aware of who is “away”.
- You can add users to a permanent list of collaborators, which is stored here in the buddy list, similarly to an instant messaging roster.
- Every time you start a session, Saros automatically starts a multi-user chat room where all participants can instant-message each other.

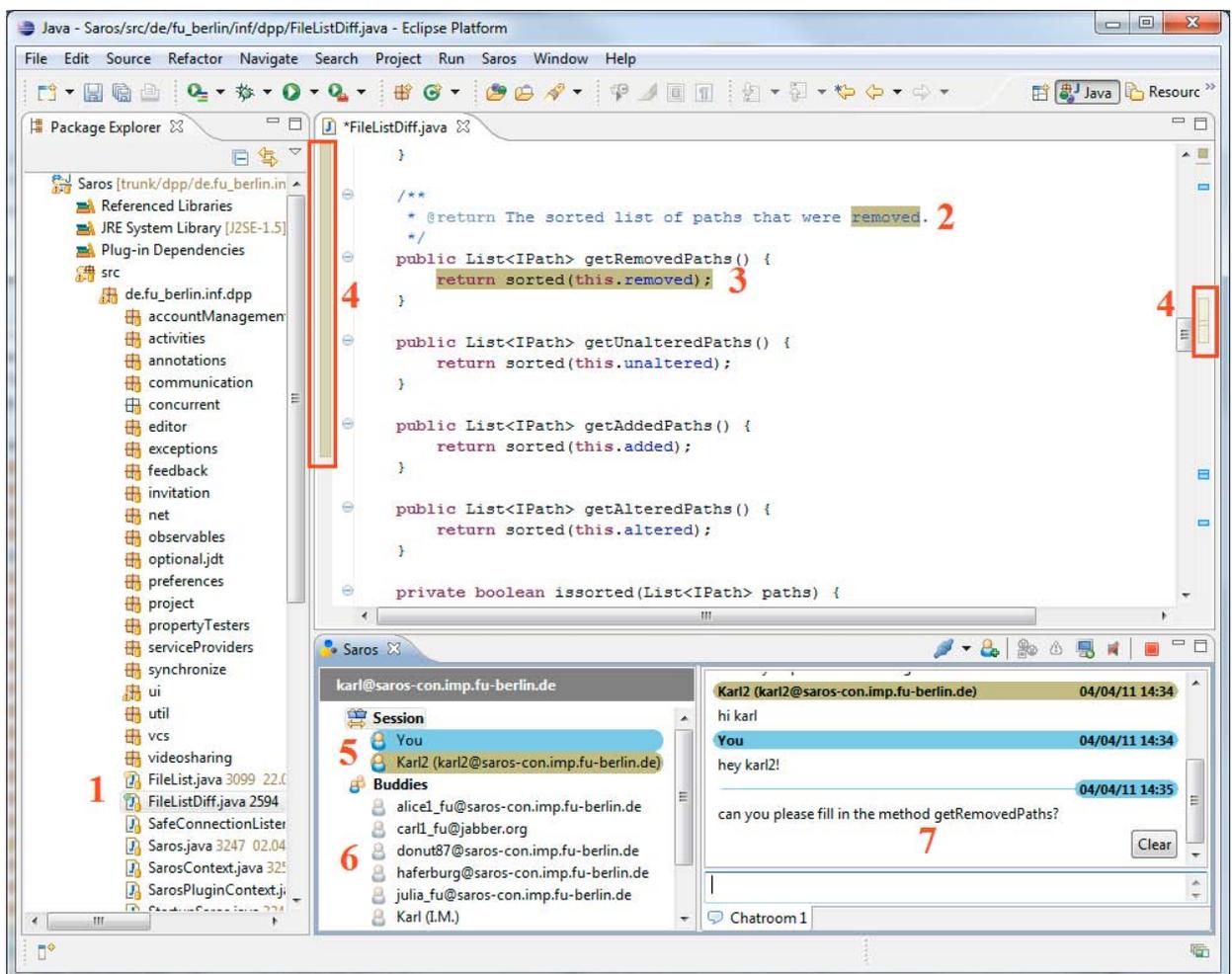


Figure 1: An annotated screenshot of an active Saros session

Throughout a session, all these things are constantly varying as everybody types, scrolls, moves between files and so on. It is actions like these that are communicated by the messages which Saros passes around to ensure that everybody's views and files are perfectly synchronised.

Additional features include:

- **Follow-mode:** Whereby you can choose to follow a user's actions; whenever they scroll or switch to another file, the resulting view is replicated on your desktop.
- **Simple user management:** Users can be granted write or read-only access.
- **VoIP:** Currently one-to-one, although we hope to have conferencing soon.
- **Distributed whiteboard:** A canvas on which users can share simple sketches and diagrams.
- **Screen-sharing:** You can send a video stream of your desktop to another user, either the whole screen or in a zoomed “camera mode” controlled by the mouse.

Scenarios

We believe that there are a number of scenarios in which using Saros can be highly beneficial:

Joint review: One participant (the “driver”) reviews the contents of one or more files together with other participants (“observers”). The observers use follow mode to watch as the driver reads. The driver can stop to raise questions (via IM or VoIP), highlighting text with the mouse to inform the others what they are questioning. Any suggested changes can immediately be discussed and implemented.

Coaching beginners: This is much like the previous scenario, except that explaining rather than reviewing is the goal. An experienced developer (the “teacher”) wishing to introduce beginners to the code takes control of the session and uses IM or VoIP to convey the important points. Each beginner follows the teacher and can raise questions when the need arises. Also, a beginner could individually peek at other regions of the file or even other files without interrupting the flow of the session for the others.

Distributed Party Programming: Two or more participants (parties) work together in a loosely coupled fashion. They work independently much of the time, but can call on one another to help whenever the need arises, possibly for just a short time. The buddy list allows everyone to see who is “in the room” and can therefore be called upon. In this mode, distributed work can even be more powerful than working in the same room, as nobody needs to leave their seat to help others and participation is not limited by whoever can fit around one computer screen.

Distributed Pair Programming: A special case of Distributed Party Programming in which two people develop code or text in continuous close collaboration, discussing the approach and combining the best of their ideas. At any one time, one person acts as a driver and the other is an observer. The observer will use the follow-mode to watch the driver.

Technical Information

Saros is build atop some key technologies discussed here.

To make the whole communication possible Saros uses XMPP (formerly called Jabber), an open standard for message-oriented networked applications. This widely-used and extensible protocol provides for Saros the messaging, roster functionality and presence information. Because it is a decentralised architecture, Saros can be used with any public XMPP server in the world (examples include jabber.org, jabber.ccc.de, as well as our own server saros-con.imp.fu-berlin.de). You can even elect to run an XMPP service on your own server and use that. Regardless, once a server has been located, the user need only create a new user profile and begin using it to run Saros sessions.

The concurrent editing capabilities of Saros are provided by the Jupiter algorithm for maintaining consistency between participants, even as they type simultaneously. The algorithm is based on the GOTO inclusion transformation, which allows each participant to write in real-time on his or her local document while remote modifications of other participants are merged into the document concurrently.

A consistency watchdog continually monitors the project to make sure it remains consistent with everyone else's copy in the session by guarding against unforeseen events that affect consistency (e.g. editing a shared file outside of Eclipse).

It is also worth mentioning that Saros includes the ability to synchronise projects by using version control system information, rather than sending *all* data between participants. In this instance, if your shared project is under version control, all participants simply share repository information and can interact directly with the repository instead. Most important actions (update, switch etc.) will be sent to all participants and replayed in order to keep the project consistent. At the moment only Subversion is supported, but we hope to include support for other revision control systems in future.

Summary

Saros is a tool that provides a wide range of functionality essential to distributed collaborative programming. It makes up for the deficiencies in working separately by automatically sharing important awareness information. It provides multiple ways to communicate with your fellow collaborators. It has a range of supporting features to replace those capabilities missing in distributed settings. This range of functionality makes it a flexible tool, applicable in a number of common software development scenarios that would otherwise be difficult or slow to carry out. Finally, it is built on technologies and standards that are well-tested and open, making it reliable foundation for your collaborations.

Acknowledgements

The author would like to acknowledge the contributions and assistance of his colleagues in the preparation of this article: Prof. Dr. Lutz Prechelt, Stephan Salinger, Christopher Özbek and Julia Schenk.

StarUML

Franco Martinig, Martinig & Associates, www.martinig.ch

StarUML is an open source software modeling tool that supports UML (Unified Modeling Language). It is based on UML version 1.4, provides eleven different types of diagram and it accepts UML 2.0 notation. It actively supports the MDA (Model Driven Architecture) approach by supporting the UML profile concept and allowing to generate code for multiple languages.

Web Site: <http://staruml.sourceforge.net/en/>

Version Tested: StarUML version 5.0.2.1570, tested on Windows XP in September 2011

System Requirements: Windows 2000, Windows XP, or higher; Microsoft Internet Explorer 5.0 or higher; 128 MB RAM (256MB recommended); 110 MB hard disc space (150MB space recommended)

License & Pricing: Open Source

Support: User mailing list

Installation

The installer follows the classic Windows install procedure without issues.

Documentation

The same help that could be browsed on the StarUML web site is available with the tool on your desktop. Documentation describes the concepts of tool but on high level vision. A more detailed documentation is available for the diagramming functions. Sample projects are provided with the tool and one of them contains the model of the tool itself, showing that the developers were able to eat their own dog food. Besides English, documentation exists in Korean, Japanese and Russian.

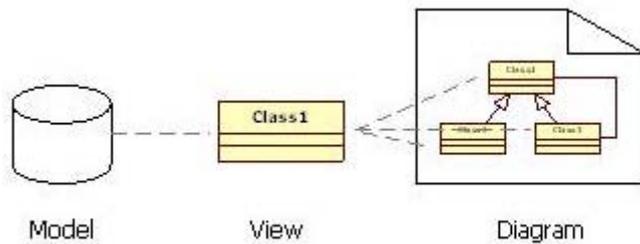
Configuration

Some general and diagram configurations options are available from the Tools/Option menu. You will find in this window also the configuration switches for the code generation. The interface is also very configurable as you can select what part of the tool you would like to view or not.

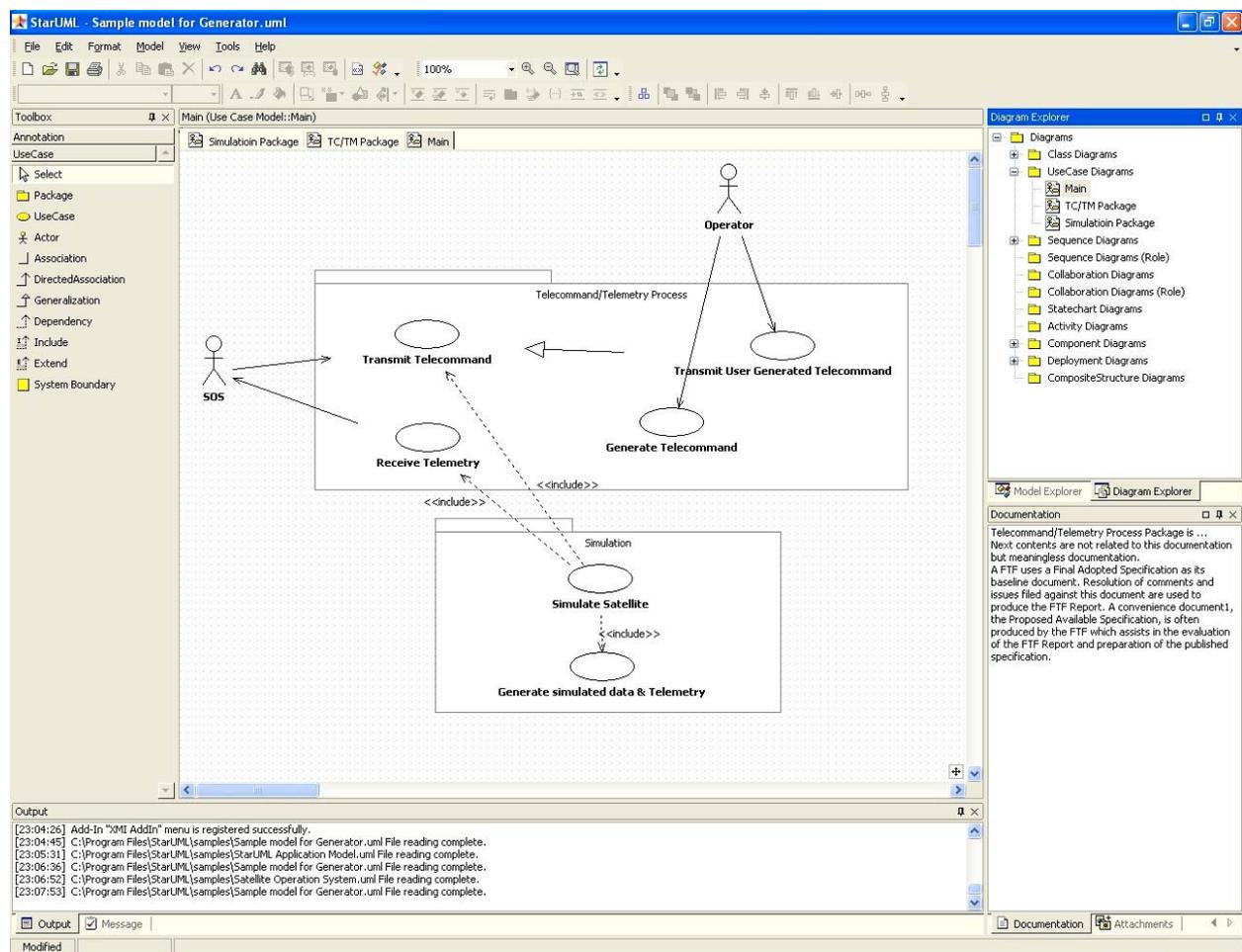
Features

When you start a new project, StarUML proposes which approach you want to use: 4+1 (Krutchen), Rational, UML components (from Cheesman and Daniels book), default or empty. Depending on the approach, profiles and/or frameworks may be included and loaded. If you don't follow a specific approach, the "empty" choice could be used. Although a project can be managed as one file, it may be convenient to divide it into many units and manage them separately if many developers are working on it together.

StarUML makes a clear conceptual distinction between models, views and diagrams. A Model is an element that contains information for a software model. A View is a visual expression of the information contained in a model, and a Diagram is a collection of view elements that represent the user's specific design thoughts.



StarUML is build as a modular and open tool. It provides frameworks for extending the functionality of the tool. It is designed to allow access to all functions of the model/meta-model and tool through COM Automation, and it provides extension of menu and option items. Also, users can create their own approaches and frameworks according to their methodologies. The tool can also be integrated with any external tools.



StarUML supports the following diagram types

- Use Case Diagram
- Class Diagram
- Sequence Diagram
- Collaboration Diagram
- Statechart Diagram
- Activity Diagram

- Component Diagram
- Deployment Diagram
- Composite Structure Diagram

The user interface is intuitive. On the upper right side, a window allows to rapidly navigate between all the content of a project, adopting either a model or a diagram view. Multiple diagrams can be open at the same time and tabs allow switching rapidly between views. The lower right window allows to document the current diagram, either with plain text or attaching an external document. During diagram editing, "wizards" are located around the object that give you the quick shortcuts to main associated tasks with your current operation, like adding an attribute when you create a class for instance. A right-click on the mouse brings the full set of operations at your disposal.

StarUML has also a model verification feature. You can export diagram in different formats (jpg, bmp, wmf). It also supports a patterns approach and import of Rational Rose files.

StarUML Generator is platform module to generate various artifacts (like as Microsoft Word, Excel, PowerPoint, and Text-based artifacts) by templates depending on UML model elements in StarUML. The users can define their own templates and can apply many different kinds of templates to the same UML model, so the users can get various artifacts automatically, easily and fast. The tool supports code generation and reverse engineering for Java, C# and C++.

Conclusion

StarUML has many powerful features and is certainly more than a "simple" diagramming tool. With its support of MDA (Model Driven Architecture), it is more aimed at people using UML in an intensive way and with some code generations objectives than for simply drawing diagrams to document requirements. However, using StarUML just as a diagramming tool work fine, especially on Windows as the tool is built with Delphi and might execute faster than the Java-based tools.

Further Reading

StarUML Tutorial

<http://www.clear.rice.edu/comp201/07-spring/info/staruml/>

A Brief Guide in Modeling UML using StarUML

<http://www.liacs.nl/~chaudron/se2010/A%20Brief%20Guide%20on%20UML%20Modeling%20with%20StarUML.pdf>

Star UML

<http://www.developeriq.in/articles/2010/mar/05/star-uml/>

Systems and Software Driven Innovation

Does software determine a larger part of your product's performance? The importance of software embedded with products is increasing at a very rapid rate! Integrity, a PTC product, is software which manages all software system development processes. With Integrity, engineering teams improve productivity and quality, streamline compliance and gain complete product visibility, which ultimately drives more innovative products into the market.

MKS is now part of PTC - <http://gurl.im/2f7e1N4>

Advertising for a new Web development tool? Looking to recruit software developers? Promoting a conference or a book? Organizing software development training? This classified section is waiting for you at the price of US \$ 30 each line. Reach more than 50'000 web-savvy software developers and project managers worldwide with a classified advertisement in Methods & Tools. Without counting the 1000s that download the issue each month without being registered and the 60'000 visitors/month of our web sites! To advertise in this section or to place a page ad simply <http://www.methodsandtools.com/advertise.php>

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 © 2011, Martinig & Associates
