Falling Blossoms

our javelin™ process

introduction

Falling Blossoms has devoted more than fifty man-years over the past decade and a half to creating and evolving its Javelin process for controlling projects.

Initially born out of a pressing need to improve the success rate of software development projects, nowadays Javelin and its related variants have proven their worth across a broad spectrum of project types, from CMMI level 5 software development projects through consulting engagements, BPR and Enterprise Engineering projects, to weddings(!) and more.

Javelin belongs to the Agile family of processes, sharing concepts and philosophies with other agile processes like Extreme Programming (XP), Scrum and Crystal. In particular, we have intentionally designed Javelin to be:

• Responsive	– Easily admits even radical change – at controlled points
• Risk-based	– Explicitly manages a wide gamut of common project risks to ensure *guaranteed* success
• People-centric	– Values individuals and interactions over processes and tools
• Lightweight	– Emphasises useful results over make-work or paperwork
• Comprehensive	– Covers most if not all areas of e.g. CMMI
• Cohesive	– Addresses *all* common software development risks (the 'all holes in the boat' principle – c.f. Gilb) Snugly interlocking practices Much greater than simply the sum of its parts
• Improvement-oriented	– Integral and explicit continual process improvement based on the Shewhart cycle (Kaizen)
• Proven and polished	– Continually used and improved in active service since 1994

e: enquiries@fallingblossoms.com
w: www.fallingblossoms.com
t: +44 (0)1474 703940

We distinguish Javelin from its siblings, however, by believing that it incorporates much best practice from various fields both present and past, rather than trying to re-invent project management from scratch. Key influences include:

- Explicit and deliberate risk management (Capers Jones, DeMarco & Lister)
- Explicit and deliberate stakeholder requirements management (Weinberg, Gilb)
- Socio-cultural aspects (DeMarco & Lister, Yourdon)
- Process Engineering (Shewhart, Deming, Shingo, Juran, Goldratt, Jacobson)
- Quality (Deming, Crosby)
- Quantitativeness (Gilb)
- Measurement (Fenton, Gilb & Graham)
- Agile (XP – Beck; Scrum – Schwaber; Crystal – Cockburn; DSDM; RUP)
- Programme Management and Theory of Constraints (Goldratt)

name

We chose the name 'Javelin' to signify that it is one of Falling Blossom's SPEAR (Software Process Engineering And Re-engineering) range of processes. We also like to think it evokes an impression of a potent instrument, light in weight, low in cost, easy to learn to use, and with a highly effective point. 'Everyone in Falling Blossoms carries their own Javelin'.

your own spear

Javelin works for *us*. But in our experience – helping organisations transition to more effective project management practices – we find that each organisation improves more quickly and achieves better results when they take the key concepts from Javelin and focus on building a process tailored to their own needs, culture and environment. Plus, a mature tailored process can become a valuable intellectual property asset for the organisation, affording significant competitive advantage over competitors' capabilities in bringing new products and services to market.

summary

To sum up, Javelin regards delivering a successful project as much like riding a moto-crosser in the mud: To go really fast you have to relax your grip: provide a gentle nudge in the general direction and you'll do fine, hold on too tight and you'll surely fall flat on your face.

javelin key concepts

risk management

Most project management processes offer ‘canned’ – or *implicit* – mitigations to common project management risks. Javelin itself incorporates many industry best-practice, lightweight, mitigations to common project risks such as:

- Building the wrong thing
- Building the thing wrong
- Failing to respond to changing circumstances and needs
- etc.

But a Javelin project team will also *explicitly* manage all the risks facing the project, to *ensure* a successful outcome for the long-suffering customer. Not only does a Javelin project continually readjust itself to keep in the ‘sweet-spot’ of delivering maximum customer value, it also continually morphs to ensure it’s always using the best approaches (e.g. processes, methods and tools) to meet those needs.

the shewhart cycle

The Shewhart Cycle, also named the Deming Cycle, the Deming Wheel, PDSA, or PDCA, represents a continuous feedback loop divided into four stages:

- **PLAN** Orient and decide what to do; consider strategy and risks; decide what to deliver; allocate available resources; etc.
- **DO** Execute against the plan
- **CHECK** Review how well we did (monitor process indicators) and decide if and how to do better next time
- **ACT** Make changes to improve the process

Javelin places the Shewhart Cycle at the heart of its cyclic approach to ***in-band*** *process improvement*.

At the start of each and every cycle (typically, of two weeks duration) the project team get together with the customer and maybe other stakeholders, to choose the key things of most immediate value to the stakeholders, plan how to deliver these things, allocate resources, and consider the risks facing the team.

Once sanctioned to proceed, the team executes the plan, producing and delivering against it. At the end of the cycle, the team come together once more to review how well things went, highlight aspects of the process that failed to work well, and suggest improvements to the process for e.g. the next cycle[Φ].

Note: Under Javelin, the team will *only* admit changes (in requirements, in the process, etc.) at the boundary *between* cycles, *never* during a cycle.

In Javelin, actually planning and implementing any process improvements gets folded into the list of things to do for the next cycle.

[□] In larger organisations, each project team will share their candidate process improvements with other project teams and/or the process improvement teams and/or the process asset library team(s).

deliverables

Javelin eschews the idea of *tasks* as the unit of planning in favour of *deliverables*. The rationale? Well, ideally we would like every cycle, every project, to meet all the stakeholders' needs with *zero* effort. Not that zero effort is a *practical* option, of course.

But we have found that placing an emphasis on *deliverables* encourages and continually reminds the team to focus on outputs (e.g. business value) rather than inputs (like, for example, hours worked). It also serves as a continual reminder to try to leverage existing components, sub-systems and solutions rather than continually re-invent the wheel.

feature schedule and backlog

At the outset of a Javelin project the team will ask the customer[Ψ] what they want, and will construct a Feature Schedule showing, roughly, the various features requested and the timeline for deliver of these features. This timeline serves to inform people outside the project team when they can expect to see various features become available to them – to help them plan, in turn.

The list of features from the Feature Schedule also feeds the Backlog – a rolling list of the features demanded by the customer, prioritised by e.g. business value. As the project moves forward, the team and customer regularly get together to select priority features (from the Backlog) for delivery in the upcoming cycle, as well as identifying additions and deletions from the Backlog.

requirements management

Unlike many agile approaches, Javelin places deliberate emphasis on the management of stakeholders’ needs and requirements.

A Javelin team will attempt to identify as many stakeholders as possible from the outset of a project, and monitor this list throughout the project. Each stakeholder, by definition, will have some needs of the project. The team tracks these (evolving) needs, from their informal beginnings, into more formal statements of requirements – both functional and non-functional.

Being an Agile process, and borrowing from the field of Lean Manufacturing, Javelin tries to keep the inventory of requirements to a bare minimum at all times, using a just-in-time approach to ensure that formal requirements become available exactly when the project team needs them – but no sooner.

people

Javelin takes to its heart the agile principle of “Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done”. Javelin explicitly includes aspects to address the needs of people and help them do the best possible job (often under ‘challenging’ circumstances!). Nothing in Javelin is *prescriptive* – we encourage project teams to question everything about the way their work works, and apply only those practices in which they find real value, whilst standing prepared to defend such decisions when challenged (or audited).

[Ψ] Well, all stakeholders, actually.

policies

Javelin promotes a very few key policies:

- Absolutely *no* work gets done off-plan. That's to say, unless a work product appears on the plan for the current cycle, no-one should spend any time working on it. Of course, if anyone has some spare time (which can and indeed should be the case in well-managed projects[α]) it might make sense to try and get ahead of the Backlog to some extent. That's what we mean by 'under-promising and over-delivering', after all.

- Although potentially capable of meeting CMMI level 5 assessment criteria, Javelin has little in the way of documentation. Next to nothing about the process is – or needs to be – written down. Its authors and guardians believe this to be a key strength – the less that is written down, the easier it is to evolve and adapt Javelin to new circumstances, applications and domains.

- Each project team has carte blanche to use some, all or none of the practices in Javelin. The only caveat is that to the extent that *the team* (not an individual) chooses to eschew a particular practice or process artefact, *the team* must be prepared to justify that decision (to e.g. process auditors, QA, management, or whoever) *and* describe their alternative arrangements for mitigating *all* the *relevant* risks addressed by the 'standard' Javelin practice or artefact.

- Each and every deliverable must meet *all* the relevant quality criteria, within the defined tolerances ('conformance to specification'!). This means that Javelin can cater for iterative development of demonstrators, prototypes and proof-of-concepts – and equally for production of industrial-quality software, systems and other process assets.

- Few projects ever have enough communication. We encourage people to get together frequently (but briefly) to exchange information and build a sense of common purpose and camaraderie. We expect everyone on the team to attend the cycle planning and review sessions, as well as contributing – in person or via e.g. tele-conferencing – to each start-the-day 'huddle' wherever possible.

[α] See e.g. Theory of Constraints (c.f. Goldratt) or Queueing Theory (c.f. Reinertsen) for an understanding of the benefits of consistently maintaining some percentage of reserve capacity.

the artefacts

In every Javelin project you will find the following 'control' artefacts. The project team evolve these as necessary to *control* the evolution and progress of the project.

project control document

Typically we lump all the following into one continually evolving monolith of a document. Some other project teams separate each item out into its own document. Yet again, some other teams use a web-based approach to sharing this information.

In any case, this collection of information serves to provide the team and stakeholders with the context we have found essential to enabling good decisions, in particular helping the team control the manifest risks that we typically find in every project:

Information	Purpose	Helps mitigate the following risks:
Project Name and Icon	Provides a sense of ownership, focus and camaraderie.	Lack of buy-in from project team. Building the wrong thing.
Article Of Understanding	Helps to foster understanding between the team and the customer.	Misunderstandings between the team and the stakeholders. Delays arising from the need to clarify requirements.
Glossary	Helps everyone to form a common frame of reference, improving communication between the team and the stakeholders.	Misunderstandings between the team and the stakeholders.
Statement of Purpose	A statement of 25 words or less; summarises and scopes the endeavour; increases a sense of ownership, focus and camaraderie; increases everyone's focus on the real goal.	Lack of buy-in from project team. Building the wrong thing.
Stakeholders and their Needs	Increases the project team's awareness of the specific needs of each key constituency.	Building the wrong thing.
Case For Action	Tracks the justification for the project; motivates and informs; increases awareness of the justification for the project; improves the basis information for disambiguating the requirements of the project.	Lack of buy-in from project team. Building the wrong thing. Delays and loss of focus from changes in personnel.
Vision	Provides everyone with a positive focus on the intended outcome; increases motivation; records the justification for the project.	Lack of buy-in from project team. Delays and loss of focus from changes in personnel. Building the wrong thing.
Risk Parade (may be held separately)	Contributes to minimisation of waste and rework; allows decision-makers to balance risk vs. reward; reduces likelihood of 'show-stoppers'; anticipates obstructions.	Project canned. Schedule and budget overruns.

Top Risks	Typically from six to ten in number; the risks the project team commit to actively managing (on behalf of the customer).	Management overheads impact delivery or quality.
Functional Requirements	Derived *directly* from the needs of the stakeholders; typically represented as Use Cases; provides the detailed engineering context for the ‘oily rags’; provides the detail necessary to manage quality effectively.	Building the wrong thing.
Non-functional requirements	Derived *directly* from the needs of the stakeholders; typically represented as Quantified Quality Objectives (c.f. Gilb); provides the detailed engineering context for the ‘oily rags’; provides the detail necessary to manage quality effectively.	Building the wrong thing.
Critical Success Factors	Typically from three to seven in number; the *most* important functional and non-functional requirements; a.k.a. ‘Top Needs’ – across *all* stakeholders; the requirements we commit to *actively* measuring and controlling.	Management overheads impact delivery or quality.
Feature Schedule and Milestones	Key synchronisation and communication tool	Management overheads impact delivery or quality.
Backlog	Key estimating and monitoring/tracking tool; allows decision-makers to continually adjust priorities, ROI; minimises inventory and work-in-progress	High levels of waste and rework.
Best Practices	Defines or refers-to pertinent best practice for building things right: Quality Plan; Risk Management Plan; Test Plan; Change Control Plan	Building the thing wrong.

cycle plan document

Each cycle plan describes the upcoming cycle: the objectives, risks, deliverables and commitment of resources.

Information	Purpose	Helps mitigate the following risks:
Cycle Plan	One to open each and every cycle; sets out exactly what's due to be delivered this cycle; sets out process improvement initiatives and deliverables; focuses the team on working on only what's wanted – by the customer; defines resource allocation and confidence ratings pertaining to the delivery of each artefact; highlights external deadlines.	Building the wrong thing. Building the thing wrong. High levels of waste and rework. Over-promising. Under-delivering. Overlooking external dependencies.
Statement of Purpose	A statement of 25 words or less; summarises and scopes the cycle; increases a sense of ownership, focus and camaraderie; increases everyone's focus on the cycle's real goal.	Lack of buy-in from project team. Building the wrong thing.
List of principal deliverables	Around 3 in number; generally references features from e.g. the Backlog and/or Feature Schedule	Building the wrong thing.
List of upcoming events	Anticipate potential obstacles and e.g. synchronisation needs of other projects	Unforeseen obstacles derail progress External demands overlooked
Risks	Focuses the project team on key risks from the project risk register that will need mitigation this cycle	Key risks remain unmitigated
Key non-functional requirements	Focuses the project team on key non-functional requirements that must be present in the deliverables of this cycle.	Poor quality
Critical Success Factors	Focuses the project team on key non-functional requirements that must be present in the deliverables of this cycle.	Poor quality Building the wrong thing
Resource Plan	To detail the allocation of resources to the production of specific work products during the cycle	Lack of resources Poor allocation of resources Overconfidence in the team's abilities (to deliver)
List of work products	The micro-deliverables and interim artefacts of the cycle; includes reviews, walkthroughs, testing, presentations, meetings, etc.; typically between 10 and 50 in number; each product is described by a set of Quality Gates, confidence ratings, and resources allocated to its completion; work products #1 and #2 of each cycle	Building the wrong thing

	always Cycle Plan and Cycle Review documents, respectively.	
Sign-off	Authorisation for commencing the cycle	Unclear acceptance criteria Building the wrong thing

cycle review document

Each cycle review provides the essential ‘closure’ for a cycle. More pragmatically, it lists both the achievements and lessons learned.

Information	Purpose	Helps mitigate the following risks:
Cycle Review	One to close each and every cycle; does not include reviews of e.g. work products from the cycle; records every learning experience when it’s still fresh in peoples’ minds; provides a regular sense of achievement, acclaim and ‘closure’; visible results, control of risk exposure, status reported in the customers’ terms.	Failing to learn key lessons. Loss of key-man expertise. Limited buy-in from project team. Premature project termination
Statement of Purpose	(Restated)	(See Cycle Plan description, above)
List of principle deliverables	(Restated); Around 3 in number Generally references features from e.g. the backlog and/or schedule Declaration of outcome (met/not met)	(See Cycle Plan description, above)
Key non-functional requirements	(Restated) Focuses the project team and stakeholders on what has been achieved this cycle.	Lack of buy-in from project team, stakeholders
Critical Success Factors	(Restated) Focuses the project team and stakeholders on what has been achieved this cycle.	Lack of buy-in from project team, stakeholders
Resource Plan	(Restated); augmented by “actuals”	Failure to learn lessons
List of work products	(actuals, status - done / not done).	Failure to learn lessons
Notes	Where a work product is ‘not done’	Lack of corporate memory
Sign-off	Authorisation for concluding the cycle; acceptance and trigger for billing.	Unclear acceptance criteria
Reservations	To allow sign-off even when not one hundred percent happy.	Lack of corporate memory Failure to complete key deliverables

functional requirements model

The functional requirements model presents a more formal, details specification of the informal functional needs of the stakeholders of the project.

Information	Purpose	Helps mitigate the following risks:
Introduction to the notation	Explains the modelling notations and conventions used in the requirements model.	Non-technical people have difficulty understanding the notation and modelling

		conventions
List of Actors	Clarifies roles and responsibilities of people engaging with the deliverables of the project.	Users may not understand how the deliverables of the project will affect them.
Functional Requirements	To specify the things the product must do; typically expressed as a Use Case Model; evolves throughout the project; populated just ahead of the construction effort (e.g. Just-in-time)	Minimises the 'inventory' of requirements. Commits minimum resource to requirements analysis Minimises the chance of requirements going 'stale'.

non-functional requirements model

The non-functional requirements model presents a more formal, details specification of the informal qualitative needs of the stakeholders of the project.

Information	Purpose	Helps mitigate the following risks:
Introduction to the notation	Explains the modelling notations and conventions used in the requirements model.	Non-technical people have difficulty understanding the notation and modelling conventions
Non-functional Requirements	To specify the things the product must do; typically expressed as a matrix of Quantified Quality Objectives (c.f. Gilb); lists all the '-ilities' of the project; examples: Cost, Timescales, Performance, Reliability, etc.; can grow to several hundred in number (eventually); evolves throughout the project; populated just ahead of the construction effort (e.g. Just-in-time). Every QQO described by: • Metric (scale, etc.). • Date-related targets: ○ Current (if known) ○ Best (ideal case) ○ Worst (worst acceptable level) ○ Planned (Planned or target level) ○ Actual (for those few actively measured)	Minimises the 'inventory' of requirements. Commits minimum resource to requirements analysis Minimises the chance of requirements going 'stale'.