10 Evolutionary Project Management (Evo) principles

This is an overview of 10 Evo process principles originally intended for participants of the Tom and Kai Gilb February 2002 training sessions with a US customer.

The intent is to allow our participants to get their mind in gear for the courses. For simplicity's sake one page per principle will explain and exemplify the principle in detail. It may be used freely with credit to Tom and Kai.

Evo Principles SUMMARY:

1. **Real results, of value to real stakeholders, will be delivered early and frequently.**
2. **The next Evo delivery step must be the one that delivers the most stakeholder value possible at that time.**
3. **Evo steps deliver the specified requirements, evolutionarily.**
4. **We cannot know all the right requirements in advance, but we can discover them more quickly by attempts to deliver real value to real stakeholders.**
5. **Evo is holistic systems engineering – all necessary aspects of the system must be complete and correct – and delivered to a real stakeholder environment – it is not only about programming – it is about customer satisfaction.**
6. **Evo projects will require an open architecture – because we are going to change project ideas as often as we need to, in order to really deliver value to our stakeholders.**
7. **The Evo project team will focus their energy, as a team, towards success in the current Evo step. They will succeed or fail in the current step, together. They will not waste energy on downstream steps until they have mastered current steps successfully**
8. **Evo is about learning from hard experience, as fast as we can – what really works, and what really delivers value. Evo is a discipline to make us confront our problems early – but which allows us to progress quickly when we really provably have got it right.**
9. **Evo leads to early, and on-time, product delivery - both because of selected early priority delivery, and because we learn to get things right early.**
10. **Evo should allow us to prove out new work processes, and get rid of bad ones early.**

If you have any questions or feedback you can:

1. Discuss the issues with colleagues
2. Look for more information about Evo at www.gilb.com (particularly the Evolutionary Project Management Course slides) and at www.malotaux.nl/nrm/Evo.
3. In fact if you have read this far, Tom and Kai would appreciate hearing from you just to see if this was worth producing for you! Drop them a line (copy in both) at tom@gilb.com and kai@gilb.com.
4. Niels@malotaux.nl would also be pleased get a cc of your comments.

Evo Principle 1:

Real results of value to real stakeholders will be delivered early and frequently

Discussion:

Real results means something that the stakeholder can use in their daily immediate work.

For example: the product is faster; easier to use; more reliable; has more useful function.

Real stakeholders are real people, both colleagues and customers, who have some interest in and use from what we are producing. By giving real results to real stakeholders we can find out in a realistic way whether they get what they want and need, whether it works as intended, and whether there are some new needs perceived as a result of the handover experience.

Early means 'next week' for many cases. It means in the first 2% of expenditure of time and money for the larger project.

Frequently means weekly or in 2% of total project time increments. It can even mean daily as in daily builds of a system being made available to in house stakeholders.

Example:

"In parallel with the development activities of the team,
selected users or customers of the system are working with and
providing feedback on the release from the previous cycle.
This feedback is used to adjust the plan for the following cycles."

Todd Cotton, HP Journal August 1996.

"The NPD team also found a range of hidden beliefs – on the part of both managers and engineers – that contributed to project cost and schedule overruns. For instance, they found that managers typically favored and funded high profile, potentially lucrative products, called 'grand slams' that required heavy investments but often failed to sell, over those based on incremental improvements on existing products that promised slow but steady growth, otherwise known as 'base hits'."

From page 101, Deone Zell, "Changing by Design: organizational Innovation at Hewlett-Packard", Cornell, 1997.

Evo Principle 2:

The next Evo delivery step must be the one that delivers the most stakeholder value possible at that time

Discussion:

This is just like a game, like chess, where of all possible moves, you want to select the 'best' one.

What is a 'best' Evo step? That depends on your current needs and values. In some cases it might be just to get something, *anything*, delivered at all. Later the focus might shift to increasing the economic value for a customer. Still later we might choose to focus on the most profitable step. So, the notion of 'best' may shift. But at any given moment we need to be clear in our project and team minds exactly what a best step is.

In Evo we can often 'compute' the best step using an Impact Estimation[1] table. The values of the stakeholders are (we hope) specified numerically so they are clear to all – in performance and in quality requirements. The project budgets for all types of resource should be equally clear and quantified. This setup (quantified requirements) gives us something with which to evaluate the several delivery step options. The option with the 'best' quantified, estimated, value-delivery, in relation to costs, should, in general, be the winner. This is very similar to chess-playing logic on a computer.

Of course, value for one stakeholder is not necessarily value for another, so a refinement of this step-priority evaluation is to decide consciously the stakeholders whose value/cost (step efficiency) we want to maximize. For example: out of our internal staff and testers; our new customers; our old customers – which one should we try to please first, and in the 'next' step?

The presentations will introduce you to the practical tools with which you can decide, as a team, which stakeholders have which precise requirements, and which Evolutionary steps deliver the best bang for the buck to the favored stakeholders.

Example:

"Evo allows the marketing department access to early deliveries, facilitating development of documentation and demonstrations.

Although this access must be given judiciously, in some markets it is absolutely necessary to start the sales cycle well before product release.

The ability of developers to respond to market changes is increased in Evo because the software is continuously evolving and the development team is thus better positioned to change a feature set or release it earlier."

Elaine L. May and Barbara A. Zimmer, HP Journal August 1996.

[1] This is one of the tools we use in connection with the Evo method. It allows us to quantify the benefits of Evo step alternatives in several dimensions of benefit, and several dimensions of cost, simultaneously. Impact Estimation Crosstalk Dec 98, "Impact Estimation Tables: Understanding Complex Technology Quantitatively"

This article can be found in its entirety on the Software Technology Support Center Web site at www.stsc.hill.af.mil/CrossTalk/crostalk.html. Go to the "Web Addition" section of the table of contents.

Impact Est article was also published in Metrics In Motion Nov98 Newsletter, visit www.distributive.com/links.htm.

Evo Principle 3:

Evo steps deliver the specified requirements, evolutionarily

Discussion:

Evo is NOT about doing a series of tasks to build a system. That is 'incremental delivery'.

Evo is about two related ideas:

- deliver real value to real stakeholders (value is defined by the requirements of those stakeholders)
- and because we have tried to deliver real value, then at each step we should measure the results!
- exactly what value was delivered?
- exactly what did it cost compared to estimates?
- did the stakeholder really get satisfied?
- did new requirements get discovered?
- did the technology work as expected?

And – what are you going to do NOW (next Evo step) about all this.

Evo is about feedback and learning. It is a 'plan → do → study → act cycle' (Deming).

One important consequence of this is that the formal requirements are very important. If they are not unambiguously clear, if benefits and costs are not quantified – then we cannot use Evo in the rational engineering mode that is the expected mode of use.

Are your most critical improvement requirements and objectives quantified, so you can engineer your system – or are they stated without numbers?

Example:

"In parallel with the development activities of the team,
selected users or customers of the system are working with and
providing feedback on the release from the previous cycle.
This feedback is used to adjust the plan for the following cycles."

Todd Cotton, HP Journal August 1996.

Evo Principle 4:

We cannot know all the right requirements in advance, but we can discover them more quickly by attempts to deliver real value to real stakeholders

Discussion:

It would be great if we could settle all requirements before we designed and implemented products. But, all experience says we cannot.

Partly this is because the customers and other stakeholders do not consciously know that they need a particular requirement. Maybe they were never asked. Maybe they don't have enough experience with the new system. Maybe they did not realize that they should articulate this as a requirement.

But the moment a real stakeholder is dealing with a real product, they can directly and indirectly voice their needs, especially if we are listening and analyzing carefully.

So – it is 'better late than never'. Evo tries as early as possible to find out about requirements we never had specified. We try to find out in practice, with real stakeholders {internal staff like testers, documentation writers, users, customers}.

Evo step delivery is in fact a powerful method for requirements analysis as well as requirements management.

Example:

	Development team	Users
Monday	• System test and release version n • Decide what to do for version n+1 • Design version n+1	
Tuesday	• Develop code	• Use version n and give feedback
Wednesday	• Develop code • Meet with users to discuss action taken regarding feedback from version n-1	• Meet with developers to discuss action taken regarding feedback from version n-1
Thursday	• Complete code	
Friday	• Test and build version n+1 • Analyze feedback from version n and decide what to do next	

Elaine L. May and Barbara A. Zimmer, HP Journal August 1996.

Evo Principle 5:

Evo is holistic systems engineering - all necessary aspects of the system must be complete and correct - and delivered to a real stakeholder environment - it is not only about programming - it is about customer satisfaction

Discussion:

Evo forces us to plan and complete *all* aspects of the product that are necessary for *all* stakeholders' satisfaction. We have to go beyond software 'functionality'; and even beyond software quality and performance. We have to consider the hardware, the training, the documentation, the testing, the help desks, the fixing process, the marketing and customer information, future extensions - *anything* that stands in the way of success if not properly dealt with.

Evo forces us to deal with these ultimate realities because every Evo step is an attempt to hand over our total product to a real environment - not usually the final 'all-customers' environment - but as real as we can make it before we finally hand over. The environment is like field trials - except we do them early (the first week of the project) and frequently (maybe 50 times in a row). If we are going to get some bad news, we'd like it as early as possible! We can also deal with a wide variety of internal and external stakeholders, one by one, on their special turf (like integration testing, bug fixing, translations, actual user learning).

Example:

This was sent to three of HP's CTO's (Chief Technical Officers) (for the overall business, as well as the imaging & printing and computer systems businesses) by Bill Crandall

Building on what we've learned from our long-term relationship with Tom Gilb, author of "Principles of Software Engineering Management,"

we believe that we've begun to understand …, with some solid research data to back it up

[, that] in particular, what matters (in terms of achieving high customer satisfaction and market success) are:

* **Early releases of the evolving product design to customers**
* **Daily incorporation of new software code and rapid feedback on design changes**
* **Teams with broad-based experience of shipping multiple projects**
* **Major investments in product architecture**

For more details, see the article from Alan MacCormack in the Winter 2001 issue of Sloan Management Review.

Evo Principle 6:

Evo projects will require an open architecture – because we are going to change project ideas as often as we need to, in order to really deliver value to our stakeholders

Discussion:

Open architecture means 'easy to change'. This can include many 'technological enablers', for many types of change. If you want to optimize your product for long term performance (and consequent survival) under conditions of change, you have to be conscious about your product architecture objectives and design specifications.

The bad news is that this is best done initially. The good news is that there are often add on interfaces for tired old legacy systems that do some useful 'opening up' for you.

We have to be flexible to respond, during a project, to the insights we get from stakeholders - the 'unexpected' requirements. We have to be able to deliver those requirements immediately without compromising system performance. That is what we need the open architecture for.

All systems need intelligent open architecture in the long run. Evo projects need it in the short term - during the project. The good news is that if you have not done the open architecture well enough - the Evo step feedback will often force you to acknowledge your poor architecture - and you will get the opportunity to improve the architecture immediately. If you were not motivated before, you will see the payoff and bite the bullet.

Better to get that great architecture *fairly* early than too late.

Example:

"Because some design issues are cheaper to resolve through experimentation than through analysis,

Evo can reduce costs by providing a structured, disciplined avenue for experimentation."

Elaine L. May and Barbara A. Zimmer, HP Journal August 1996.

"During December, detailed coding of the individual modules started. But the IE3 team was still making <u>decisions about the overall product architecture</u> - decisions that would not only affect the features in the final product but also the development process itself. A team member explained, "We had a large number of people who would have to work in parallel to meet the target ship date. We therefore had to <u>develop an architecture</u> where we could have separate component teams feed into the product. Not all of these teams were necessarily inside the company. The <u>investment in architectural design</u> was therefore critical. In fact, if someone asked what the <u>most successful aspect of IE3</u> was, I would say it was the job we did in 'componentizing' the product."

Alan MacCormack, *How Internet Companies Build Software*, MIT SLOAN Management Review, Winter 2001, about Internet Explorer 3.0 development.

Evo Principle 7:

The Evo project team will focus their energy, as a team, towards success in the current Evo step. They will succeed or fail in the current step, together. They will not waste energy on downstream steps until they have mastered current steps successfully

Discussion:

Evo demands interdisciplinary teamwork. All aspects of development must be fused continuously and frequently in order to deliver useful value to stakeholders at an Evo step. The team must learn how to deliver successfully 'in the small'. Bad ideas need to get dropped early - and cannot permeate the entire project. Authority is really pushed down to the team level. You don't need approval for things that might be large-scale risks for the entire project. You need to prove they work early in practice and continuously. If they don't, you need immediate inter-step changes to something that *does* work for real.

Example:

- **"Many development teams lack a well-defined, efficient decision-making process. Often they make decisions implicitly within a limited context,**
  - **risking the compromise of the broader project goals and slowing progress dramatically.**
- **Evolutionary Development forces many decisions to be made explicitly in an organized way,**
  - **because feedback on the product is received regularly**
  - **and schedules must be updated for each implementation cycle.**
- **The continual stream of information that the project team receives must be translated into three categories of decisions:**
  - **changes to the product as it is currently implemented,**
  - **changes to the plan that will further the product implementation, and**
  - **changes to the development process used to develop the product.**
- **Fortunately, because of Evo's short cycle time, teams**
  - **have many opportunities to assess the results of decisions and adjust accordingly"**

Todd Cotton, HP Journal August 1996.

Evo Principle 8:

Evo is about learning from hard experience, as fast as we can - what really works, and what really delivers value

Evo is a discipline to make us confront our problems early - but which allows us to progress quickly when we really provably have got it right

Discussion:

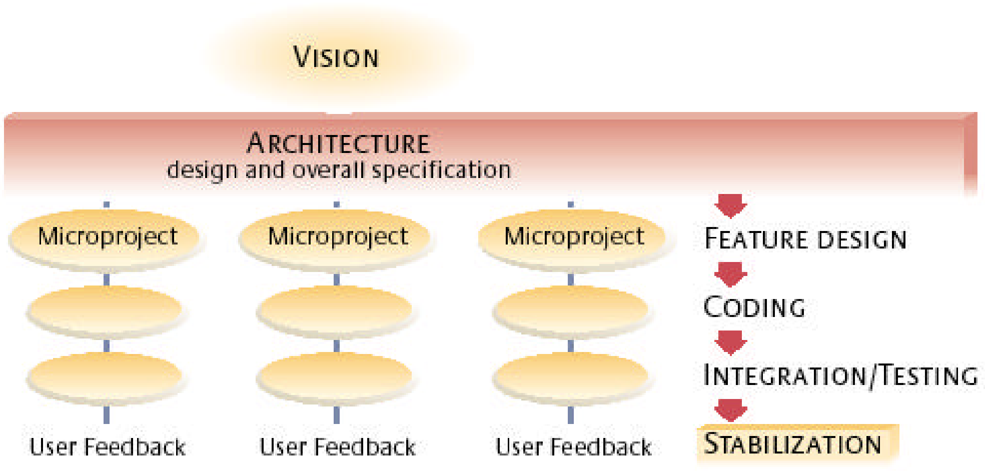
Evo may seem 'slow'. But it is the fastest known way to get the right stuff delivered. This is especially true when you are pushing the envelope with new competitive ideas that are not yet proven in practice. Evo becomes your workbench and experimental lab. 'Bill and Dave' are curious to see how things are going between each Evo step. They expect working systems on the workbench continuously.

Example:

One of the criteria commonly used in setting priorities during this initial planning activity is:

Features with greatest risk. The most common criterion used for prioritizing the development phase implementation cycles is risk. When adopting object technology, many teams are concerned that the system performance will not be adequate. Ease-of-use is another common risk for a project. The use scenarios that will provide the best insight into areas of greatest risk should be scheduled for implementation as early as possible.

Todd Cotton, HP Journal August 1996.



Evo Principle 9:

Evo leads to early, and on-time, product delivery - both because of selected early priority delivery, and because we learn to get things right early

Discussion:

Evo allow us consciously to select things for *early* and *complete* delivery. These things cannot be 'late'. We also can select high risk items for early Evo steps because then we have time to deal with the risks, and to remove them from future steps.

Example:

- **Obtaining early market and technical feedback results in increased feature evolution and customer satisfaction. (H1)**
  - **but this increases schedule estimation error. (H2)**
  - **but early market feedback increases bugginess. (H4)**
- **High-level architecture specification provides for more flexible product development measured in terms of feature evolution. (H7),**
- **Evolutionary development allows flexibility in product development allowing the project team to make requirements, functional changes and add code for new features late into the project. (H8)**
- **Design reviews identify any consistency problems earlier than the later testing activities that require a running product (H12)**
- **Running regression tests, each time developers check changed or new code into the project build, improves product quality. (H14)**
- **If the project team has enough time for final product stabilization phase, then they have completed the project on time. This results in lower schedule estimation error. (H20)**

The project team may decide to spend time on final product stabilization versus making late design changes that incorporate market and technical feedback. This may result in increased % of original features implemented in the final product. (H21)

Hypothesis supported by research on HP Projects, Bill Crandall (HP) based on Sharma Upadhyayula, 2001, MIT Master of Science [Engineering and Management] Thesis on HP Evo Project data.

Evo Principle 10:

Evo should allow us to prove out new work processes, and get rid of bad ones early

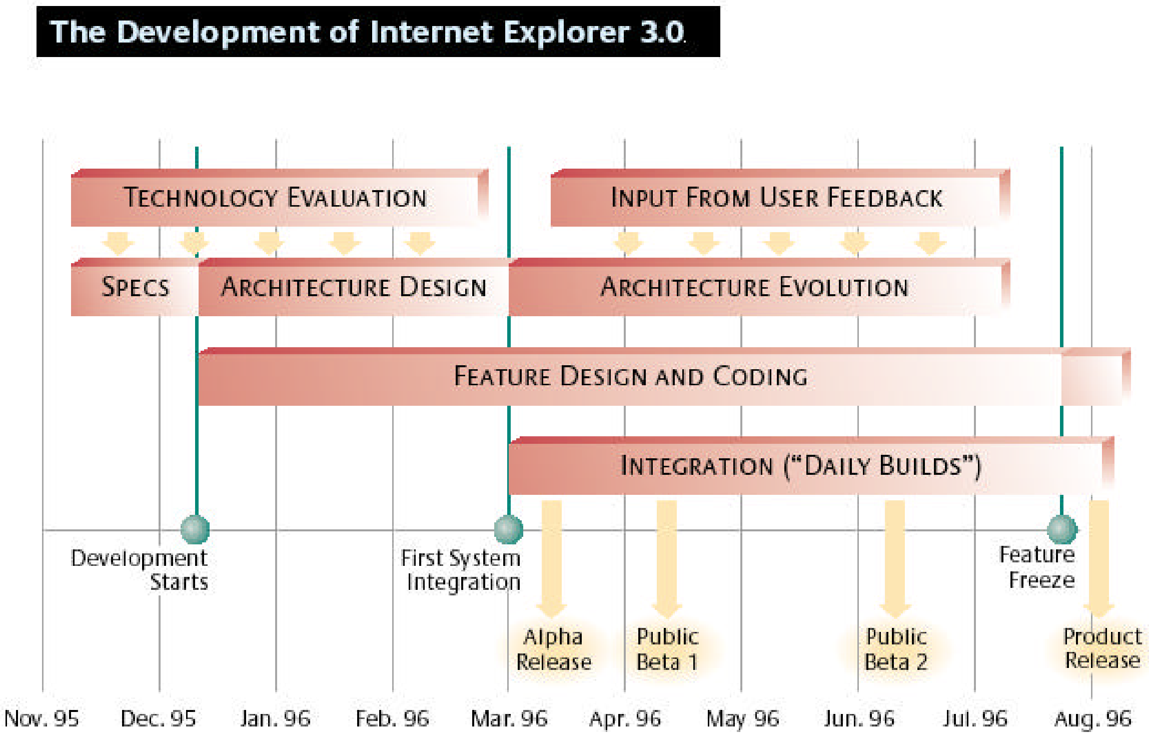
Discussion:

Evo has one advantage for people who are sceptical towards it as a new development process. It either works immediately or you can remain sceptical! But Evo can prove out old processes that might not fit well with Evo (like full regression testing at each step). Evo can try out new processes, and help you tune them to work correctly, while you are still at early stages of your project (such as Inspection in a sampling mode).

Example:

"But the IE3 team was still making decisions about the overall product architecture - decisions that would not only affect the features in the final product but also <u>the development process itself."</u>

Alan MacCormack, *How Internet Companies Build Software*, MIT SLOAN Management Review, Winter 2001, about Internet Explorer 3.0 development.



Feedback routes:

If you have any questions or feedback you can:

1. Discuss the issues with colleagues
2. Look for more information about Evo at www.gilb.com (particularly the Evolutionary Project Management Course slides) and at www.malotaux.nl/nrm/Evo.
3. In fact if you have read this far, Tom and Kai would appreciate hearing from you just to see if this was worth producing for you! Drop them a line (copy in both) at tom@gilb.com and kai@gilb.com.
4. Niels@malotaux.nl would also be pleased get a cc of your comments.