Agile Development Methods
Software Projects: Seeking Better Ways

Roy Morien
Research Fellow
DEBII, Curtin Business School
Curtin University, Perth, Australia
&
Visiting IS Specialist
Naresuan University
Phitsanulok, Thailand
Agile Development

The Nature of Software Development
The Nature of Software Development

- Software Development has been considered an engineering discipline – viz: Software Engineering
- But is it?
- McConnell (1996) suggests 4 dimensions of system development:
  - People
  - Tools
  - Process
  - Product

A development activity is done by PEOPLE (for PEOPLE), using development TOOLS, while following a PROCESS, to produce a PRODUCT.
The Nature of Software Development
It’s about the People!!

... we now know with certainty that peopleware issues have more impact on software productivity and software quality than any other factor

(McConnell p.12)

... it is now crystal clear that any organization that’s serious about improving productivity should look first to the peopleware issues of motivation, teamwork, staff selection and training.

(McConnell p.13)

See also:


The Nature of Software Development
It’s about the People!!

- “implementing systems is difficult and many projects fail. They may fail for a number of reasons. The team of people involved may prove dysfunctional. The intended users of the system may refuse to accept the change that goes along with it. Key errors may occur in the planning (done by People) or requirements gathering (done by People) stage and so on

- * my included comment.

http://exonous.typepad.com/mis/2004/02/systems_develop.html University of Prince Edward Island
The Nature of Software Development
It’s About the People

- The Project Owners / Financiers
- The Domain Experts / Users
- The Developers / Technical People
- The Project Manager

They must be able to communicate effectively.

People learn from experience – software development is a learning activity carried out by people.

Consider the lines of communication, and the learning environment:

- Requirements determination is a learning activity
- As the project proceeds, people learn.
- Knowledge must be shared – people learn from other people.
The Nature of Software Development
It’s About the People

- Software Development is an activity carried out by People, who must communicate, learn and share their knowledge.
- As people DO, so they Learn.
- So software development is a learning activity in every aspect.
- Software development should be seen as a knowledge-based activity, not a process-based activity.
- Any software development process, and software project management process, must enable and magnify the communication, learning and knowledge sharing.
The Nature of Software Development
Managing the People

- ... In the wholeheartedly reengineered corporation, responsibility and authority are so widely distributed throughout the organization that virtually everyone becomes a manager, if only of his or her own work. (Champy, p.70)

- ... The HP Way is really our core values, our company culture based on trust and respect for individuals. It’s about empowering people at the lowest level of the organization to run with their ideas. That freedom fosters a lot of creativity and enthusiasm. (Champy, p.70)

- We just started giving out assignments and trusting people to carry them through. (Champy, p.70)

- We are all in this to build a sustained competitive advantage. You are not here solely to perform your function. You are here to add value. (Champy, p.73)

The Nature of Software Development
Managing the People

- You must have a culture that encourages qualities like relentless pursuit (to match our customers’ elusiveness), bottomless resources of imagination (to create needs our customers may not know they have), and both smooth teamwork and individual autonomy (to match their demanding standards).

  (Champy, p.76)

- … You cannot have a culture of obedience to chains of command and the job slot. It just won’t work.

  (Champy, p.76)

- … enabling (people); redesigning work so that people can exercise their skills and capabilities to the fullest extent possible – then stepping back and letting it happen.

  (Champy, p.115)
The Nature of Software Development Motivating the People

- Companies die because their managers ... forget that their organizations’ true nature is that of a community of humans (de Geus, 1997)
- The amount that people care, trust and engage themselves at work has not only a direct effect on the bottom line, but the most direct effect, of any factor, on your company’s expected lifespan. (de Geus, 1997)

The Nature of Software Development Enabling the People

- The Walmart – Hurricane Katrina Example
  - ... Wal-Mart granted wide latitude to its local and regional managers in providing disaster aid.
  - In one Louisiana town, an employee drove a forklift through the door of his Wal-Mart store, to obtain bottled water for a local nursing home.
  - Another outlet became a temporary barracks for police officers whose homes had been flooded by the storm.

http://formerspook.blogspot.com/2008/03/wal-mart-to-rescue.html
The Nature of Software Development Enabling the People

- The Walmart – Hurricane Katrina Example
- ... But perhaps the best example of Wal-Mart’s empowerment policy occurred in Waveland, Mississippi, a community that was heavily damaged by the hurricane.
- Unable to reach her superiors, assistant store manager Jessica Lewis decided to run a bulldozer through her store, collect essential supplies that weren’t water-damaged. The supplies were then stacked in the parking lot and given away to local residents.
- Ms. Lewis also broke into the store’s pharmacy locker to supply critical drugs to a local hospital.

THE MAIN POINT is the lower level, local managers and staff acted on their own initiative in a positive manner without management guidance, planning or approval ... THEY CAN DO IT!

http://formerspook.blogspot.com/2008/03/wal-mart-to-rescue.html
The Nature of Software Development Motivating the People

- Is a human activity (done by people).
- The people have different levels of training, expertise, knowledge.
- Requires continual, effective communication.
- Continuous learning. Software development is a learning activity.
- Is a knowledge sharing activity, not a process driven activity.
- Software development is complex, experimental, experiential, adaptive.
The Nature of Software Development
Motivating the People

- Maintaining a qualified IS workforce has been identified as one of the top ten concerns of IS executives recently (and similar human resources issues were among top concerns in prior key issue studies over the last two decades.

The Nature of Software Development
Motivating the People

- We might look to Maslowes’ Hierarchy of Needs to think about how to motivate IS professionals
  - Physiological Needs
  - Safety Needs
  - Needs of Love, Affection and Belongingness
  - Needs for Esteem
  - Needs for Self-Actualization

That is, we should be looking in the management literature instead of the project management literature
Agile Development – Lessons from Successful Enterprises

I want now to look elsewhere in what might be called ‘process improvement’ endeavours of highly successful enterprises and projects.

Process improvement has been, in my view (arguable, I will admit) based primarily on the publication of process auditing requirements. Proof of process adherence in the way of reports, documentation and verification of conformity to an adopted process seems to be the hallmark of ‘process improvement’.

Argue with me if you will … but later 😊

Process improvement stipulations must encompass the social and behavioural aspects of ‘human activity systems’, and view processes as learning activities, and knowledge sharing.
Toyota takes world sales lead from GM
Japanese automaker sold 2.41 million vehicles in first quarter

Associated Press
updated 4:48 p.m. ET April 23, 2008

DETROIT - Toyota took the global sales lead...
Why talk about a motor vehicle manufacturing company? And why talk about this one?

- Toyota is the largest motor vehicle manufacturing company in the world, by market capitalisation.
- In its 2002-2003 financial year, Toyota made a profit of US$8.5 billion ... larger than the combined profit of Ford and GM ... the biggest annual profit of any vehicle manufacturer in the last decade.
- Net profit margins are 8.3 times higher than the industry average.
Toyota: The ‘Lean Manufacturing’ Company

- Toyota’s stock price rose 24% in 2002 … market capitalization was higher than the combined market capitalization of Ford, GM and Chrysler.
- Return on assets is 8 times higher than industry average.
- Has made a profit every year for the last 25 years.
- By 2005, Toyota was outselling every other car manufacturer in the world.
- Toyota opened new manufacturing plants in the US when all the US manufacturers were closing plants and going off-shore.

Toyota: The ‘Lean Manufacturing’ Company

- Toyota is renowned as the company that created Lean Manufacturing.
- Toyota is managed according to 14 basic management principles that can be summarised as:
  - Fostering an atmosphere of continuous learning and improvement
  - Satisfying customers (and eliminating waste)
  - Quality first and consistently
  - Grooming leaders from within the organisation
  - Teaching employees to become problem solvers
  - Growing together with suppliers and partners for mutual benefit.
LEADERSHIP – NOT MANAGEMENT: A SIGNIFICANT SUCCESS FACTOR
Team New Zealand

- In 1995, a team from New Zealand won the famous and prestigious yacht trophy, called the Americas Cup. This was only the 2nd time in 146 years that a non-US syndicate had won the trophy … Australia had won it once before.

- The amazing thing was that the NZ yacht won 41 of the 42 races that they competed in over the 6 months competitive campaign. What was even more amazing was that the NZ syndicate had a very limited budget, and a limited amount of time to develop their record-winning boat.

Team New Zealand

○ How did they do it? There are some valuable lessons here in this experience and success that are very applicable to software development.

○ The success has been attributed to:
  ● The inspirational leadership of the syndicate Leader
  ● The strong sense of community within the syndicate team
  ● The openness of communication between team members
  ● ‘Customer’- led development – the sailors!!!
  ● The sustained rate of continual improvement (of the boat speed)
  ● The level of commitment and purpose by all participants

○ This syndicate exhibited many of the valuable traits of a ‘learning organisation’:
The Success Factors

**TOYOTA**
- Fostering an atmosphere of continuous learning and improvement
- Satisfying customers (and eliminating waste)
- Quality first and consistently
- Grooming leaders from within the organisation
- Teaching employees to become problem solvers
- Growing together with suppliers and partners for mutual benefit.

**TEAM NEW ZEALAND**
- The strong sense of community within the syndicate team
- The openness of communication between team members
- ‘Customer’-led development – the sailors!!!
- The sustained rate of continual improvement (of the boat speed)
- The inspirational leadership of the syndicate Leader
- The level of commitment and purpose by all participants
The Learning Organisation

“The function of the society of post-capitalist organisations ... is to put knowledge to work ... *it must be organised for constant change*”

Peter F. Drucker

The Core Capabilities of a Learning Organisation:

1. Creative orientation
2. Generative discussion
3. Systems perspective

(Maani & Cavana, p138.)
The Learning Organisation

- Creative orientation
  
  *The source of a genuine desire to excel. .. The source of an intrinsic motivation and drive to achieve ... favours the common good over personal gains.*

- Generative discussion
  
  *A deep and meaningful dialogue to create unity of thought and action*

- Systems perspective
  
  *The ability to see things holistically by understanding the connectedness between parts.*
The Learning Organisation

The 5 Core Disciplines of a Learning Organisation:

1. Personal Mastery
2. Shared Vision
3. Mental Models
4. Team Learning and Dialogue
5. Systems Thinking
The Learning Organisation

The 5 Core Disciplines of a Learning Organisation:

- **Personal Mastery**

  *Instills a genuine desire to do well ... focus on the desired result, not the process itself ... requires a commitment to truth – continually challenge current practices and norms (if not, can distort reality and prevent knowing where they really stand.)*

- **Shared Vision**

  *Shared vision aligns diverse views and feelings into a unified focus ... management cannot impose a shared vision ... a sense of community and shared vision is essential for long-term survival.*
The Learning Organisation

The 5 Core Disciplines of a Learning Organisation:

- **Mental Models**
  
  *The first step in any change process is to unfreeze the present patterns of behaviour and thinking as a way of managing resistance to change. The leader has a pivotal role in dismantling negative mental models and shaping new ones.*

- **Team Learning and Dialogue**
  
  *Dialogue is an essential requirement for organisational learning ... constructive feedback ... mutual encouragement among group members is essential ... a knowledge sharing environment ... a learning environment ... ideas are important, not the people who suggest the ideas ... egoless behaviour.*
The Nature of Software Development
What is the Client really buying?

- Not buying a product, but buying a development activity.
- Software development is product development.
- It is more aligned to research than to production or manufacturing.
- Do we do research in distinct activity phases, serially undertaken?
- Product development requires creative thinking, experimentation.
- Can you ‘command and control’ creative thinking & experimentation?
- Software development requires the continued and authoritative involvement of the client in developing the software;
The Nature of Software Development
Can your IS development people be trusted?

- Management of professional people cannot be ‘command and control’.
- You cannot impose a requirement “Be creative, productive and adhere to the plan” (the plan decided by someone else)
- Professional people work best without direct supervision, finding their own way to be effective, efficient, creative and successful in doing their job.
- **Can your IS development people be trusted?**
- If not, then the university has failed to give them a proper education.
- If not, then your organisation has failed to have proper and effective selection processes.
- If not, then you have failed to provide proper and adequate training.
- If not, the organisation has a negative and demotivating culture, and management style.
- You have a failure to educate, select, train and motivate!!
Agile Development

The Assumptions of Today
The Assumptions of Today

Traditionally software development processes have been predicated on the following assumptions:

- It is possible, efficient and effective to create a thorough and complete analysis of requirements for a system, at the start of the project.
- It is possible to plan in detail the development project, even for 2-3 years ahead.
- It is possible to provide accurate estimates of the cost of the project of the development time, and the completion date.
- Stable requirements will remain unchanged during the project period.
- Following the plan is the best way to ensure project success.
The Assumptions of Today

- A serial or linear **phased development** approach is the best, most efficient, most effective approach to software development.
- Software development is fundamentally a **technical activity** that can be planned and managed according to a preconceived plan.
- Software projects can be undertaken in the same way as civil **engineering** projects, which is orderly, linear and sequential.
- Certainty of outcomes can be achieved if only we properly, correctly and comprehensively plan, analyse, estimate, document and manage the project.
- Project success is counted in terms of ‘Within Budget, Within Time, Within Scope’.
The Assumptions of Today

On every count, these assumptions have been demonstrated to be WRONG.
The Problem

- Business Systems Development over the last 30 years has been far from successful.

- $250 BILLION is spent on 175,000 I.T. apps each year in the USA alone
- $140 BILLION is wasted due to lack of process and standards
- $110 BILLION is the value we are getting -- less than half of our total investment!!

Source: Standish Group, 1995
The Problem

Achieved Features... only 61%!

*Source: Standish Group, Inc. (Application Development Trends • January 1995)*
% OF Systems Projects Considered to have been Successful, Failed, or were Challenged as to Outcomes

Succeeded  Failed  Challenged
The Problem

Project Outcomes

- System used after changes: 3%
- System used as delivered: 2%
- System used but later abandoned: 20%
- System paid for but never delivered: 28%
- System delivered but never used: 47%
The Problem

- The Price of Failure (*)
  - US$125 million failure of the CONFIRM Travel Reservation System.
  - US$45 failure of the Californian Dept of Motor Vehicles Project
  - Florida’s bungled attempt at developing a state benefits system (see later for further discussion)
  - UK£16.5 million failure to computerise the Inland Revenue returns system
  - The London Stock Exchange automated settlements system ~ £500 million lost.
  - OK Dept. of Social Security: 17 years in development, deemed over budget by 400%

The Problem

- And in Australia
  - Wespac Bank in the 1980s ... CS90’s Project ... $200-$300 million written off with the project failure.
  - Australian Blood Management System - $4 million budget blew out to $40 million, and 4 years late.
  - New Customs IT cargo system fails to deliver:

  THE failure of the new integrated cargo system implemented by the Customs Service a week ago has produced "a catastrophe" on the wharves of Melbourne, Sydney, Brisbane and Perth
The Problem

- New Customs IT cargo system fails to deliver
- By Garry Barker
  Technology Editor
  October 21, 2005

- The new and needed update of the Customs computer system was originally costed at $33 million. The cost had blown out to more than $250 million, and the system still did not work,

- It was supposed to take us into the new millennium. It is three years late in roll-out.

- "We said in September we were scared witless it wouldn't work," he said. "We were told we were doomsayers. We have been told that as late as October 11 the system was not in a position to be cut over, but they did it anyway."

- "the most serious crisis to hit this industry since our council was formed in 1904".
The Cause of the Problem?

The Waterfall (SDLC) Model

1. Requirements Definition
2. System & Software Design
3. Implementation & Unit Testing
4. Integration & System Testing
5. Operation & Maintenance

Disappointment & Rejection (in about 72% of cases in 2000)
The ‘agile developers’ view of the Waterfall Approach:

- It creates the highest risk, at the start, when you try to plan far ahead ... the plan must be adaptive to acknowledge ‘future uncertainty’.
- It assumes that there is nothing more to learn after the initial analysis phase, therefore rejecting any future learning.
- Requirements are perishable ... they become stale and irrelevant as they sit ‘on the shelf’ waiting to be developed. Just-in-Time Analysis and Design is more successful.
- Distant completion deadlines are not drivers of effort.
- The phased approach is unnatural ... Analysts also Design, Designers also Analyse, Programmers always Analyse and Design
- After the Analysis Phase, the clients and users become irrelevant. Here is no meaningful collaboration.
The ‘agile developers’ view of the Waterfall Approach:

- After the Analysis Phase, the clients and users become irrelevant. Here is no meaningful collaboration.
- There is no realistic feedback cycle.
- Change is at best ‘managed’, at worst refused.
- Reliance on penalties to enforce contractual agreements.
- Lack of transparency of progress.
- End of Line QA is ineffective.
- Lack of focus results in errors.
- Inability of the user to verify the requirements document.
- No realistic validation of requirements until implementation ... too late!!

Significant project RISK is created at the start of the project
The Assumptions of Today

- Up-Front, comprehensive analysis is not possible.
- Without a plan we will fail, but when the battle begins, the plan is useless – ADAPTIVE PLANNING is required.
- An estimate is inherently inaccurate, and the longer the timeframe, or the greater the complexity of the system, the more inaccurate the estimate will be. Uncertainty is an inherent characteristic of ‘the future’. Continuous estimating is necessary.
- Requirements invariable and inherently change, whether we try to ignore the changes, or not. Welcome Change ... because every change is a step towards a needed and useful system.
- Software development is a chaordic activity, and a human activity, that cannot be managed according to a preconceived plan, but demands an adaptive, empirical approach for the plans and estimates to remain relevant.
- Command and Control management of professional, competent software developers is neither necessary nor successful.
The Assumptions of Today

- A phased development approach is wasteful and futile, and flies in the face of system thinking and process theory. It is not appropriate to the nature of software development.
- Software development is fundamentally a human activity that demands creativity, adaptability and a collaborative and cooperative environment (or ecosystem) to be successful.
- Software development is a learning and educational experience, and is more knowledge based than process based.
- Software development is ‘product development’, not ‘product manufacturing’. We are developing ‘soft’ware that is inherently changeable and malleable, unlike civil engineering projects which result in ‘hard’ware (in fact, concrete and steel ‘ware’ which is inherently difficult and costly, if not impossible, to change).
- Certainty of outcomes can be only be achieved by constant inspection, verification and validation by all parties to the development, in a Just-in-Time fashion.
- Project success is counted in terms of ‘business value’.
Research Evidence

- Thomas, M., 2001 IT Projects Sink or Swim, British Computer Society Review.

- 1027 Projects studied for “failure factors”, 87% failed, scope management related to attempting waterfall practices (including detailed up-front requirements) was the single largest contributing factor for failure – cited in 82% of the projects as the #1 failure factor.

- “The approach of full requirements definition followed by a long gap before those requirements are delivered is no longer appropriate”

- “… any assumption that there will be little significant change to requirements once they are documented is fundamentally flawed, and that spending significant time and effort defining them to the maximum level is inappropriate”
Research Evidence

- Productivity Issues


- Indicated that productivity as measured in function points was nearly 20% higher per developer for iterative, incremental development, over a developer using the Waterfall Model.

- The study also showed that the less “rigorous” the adherence to the Waterfall Model, the higher the productivity, and

- The more “iterative, incremental” the “agile” approach, the higher the productivity.
Agile Development &
Agile Project Management
Words to Understand

- Iteration, Iterative
- Incremental, Increment
- Evolutionary
- Adaptive
- Emergent
- Just-in-Time Requirements Determination
- Collaboration
- Retrospective (Review)
- Chaordic
The Agile Manifesto

states the following guiding principles:
We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

<table>
<thead>
<tr>
<th>WE Prefer</th>
<th>over</th>
</tr>
</thead>
<tbody>
<tr>
<td>Individuals and interactions</td>
<td>processes and tools</td>
</tr>
<tr>
<td>Working software</td>
<td>comprehensive documentation</td>
</tr>
<tr>
<td>Customer collaboration</td>
<td>contract negotiation</td>
</tr>
<tr>
<td>Responding to change</td>
<td>following a plan</td>
</tr>
</tbody>
</table>

That is, while there is value in the items on the right, we value the items on the left more.

Agile Development is …

- “human centric bodies of practices and guidelines for building usable software in unpredictable, highly-volatile environments”. Software development projects are considered to be an unpredictable and highly volatile environment.


- “a departure from plan-driven traditional approaches, where the focus is on generating early releases of working software using collaborative techniques, code refactoring, and on-site customer involvement”.

Agile Development is …

“A software development method is said to be an agile software development method when a method is people focused, communications-oriented, flexible (ready to adapt to expected or unexpected change at any time), speedy (encourages rapid and iterative development of the product in small releases), lean (focuses on shortening timeframe and cost and on improved quality), responsive (reacts appropriately to expected and unexpected changes), and learning (focuses on improvement during and after product development)”

Agile Development is …

- **People Focused:** (1) Collaborative: collaboration between developers and clients is continuous and continual. (2) Self-Organising and Self-Managing Teams: Significant responsibility is handed to the team members, rather than the Project Manager, to decide on the work to be done in the next iteration.

- **Empirical and Adaptive:** Project management practices that have been published to support ‘agile development’ practices are described as ‘empirical’, ‘adaptive’, ‘evolutionary’ or ‘experiential’ rather than ‘prescriptive’, or ‘pre-planned’.

- **Iterative:** Development is achieved through a series of short iterations each of which produces a useable enhancement to the system.
Agile Development is …

- **Incremental**: Development is achieved through a series of delivered increments to the system, each of which produces a fully developed, fully tested and certified extra feature or component of the system.

- **Evolutionary**: the system grows in size, the requirements *in detail* are continuously discovered, and are continually emergent during the development period.

- **Just-in-Time Requirements Elicitation**: Requirements are stated in detail ‘just in time’ to develop them, in the iteration in which those requirements will be implemented.

- **Knowledge-Based**: Development activity is decided upon by the knowledgeable, self-managing members of the team, with continual knowledge sharing about the product, the technology and the progress of the project.
Project Planning

The Iterative Project Plan

Every iteration creates and delivers working software to the Client

The same activity cycle is repeated in every iteration
The Student Information System

I want information on ...

I want to be able to find ...

Students Information

Staff Information

Class Timetables

Courses, Subjects

Class Rooms

The Universe of Discourse
Iteration Cycle Activity

The Iteration Plan

Plan

DO

REVIEW

Yes!! Deliver!!
Deliver working, fully tested, useful code to the Client

DELIVER
ER Based Development Cycle

ITERATIVE, INCREMENTAL DATABASE DEVELOPMENT

The database design grows, iteration by iteration

Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
Iteration n

The same activity cycle is repeated in every iteration
The Iteration Activity Cycle

- Entity Identification & Definition
- Deliver to Client
- Construct Reports
- Table Definition
- Table Construction
- Table Lookup Form Construction
- Table Maintenance Form Construction

The Focal Entity Prototyping Iteration Cycle
User Story Driven Development

Finding and Analysing User Stories

User Stories

User Interviews

List of User Stories (Product Backlog)

User Story includes Database Requirements

User Story includes Processing Requirements

Processing Requirements include Database Requirements

Database Requirements include Processing Requirements

Database Requirements create Processing Requirements
Project Estimating

The Barrier to Acceptance
The question hanging over ‘agile development’ always seems to be ‘But what about project estimates? Clients (directors, financial officers …) want an exact estimate up front, before they commit to the project’.
Estimating: The Barrier to Acceptance

My immediate thought about this is ‘Let’s get real!!’

1. an estimate is just that … an estimate. Too often it is taken by the client to be a quote.

2. An estimate is someone’s best guess at the future activities and outcomes of the project.

3. The longer the time period being estimated, the greater the margin of error and uncertainty

4. The bigger and more complex the project, the greater the margin of error and uncertainty.

5. Uncertainty is real … attempts to eliminate uncertainty are bogus and humbug.
Estimating: The Barrier to Acceptance


“It is remarkable that in our computer industry one third to one half of its effort and cost is out of control”. “...two hundred to three hundred percent cost overruns and up to one hundred percent time slippages have been common, frequent, almost universal”.


“...more projects are doomed by poor cost and schedule estimates than by technical, political or team problems.”
Estimating: The Barrier to Acceptance


“...53% of projects overrun cost estimates by 189% or more (at a cost of US$59 billion per year in the US alone)”

This research is supported by a US government study on software development projects, which revealed that 60% of projects were behind schedule and 50% were over cost.
Estimating: The Barrier to Acceptance

The Cone of Uncertainty

Estimating: A Brief Note About Project Failure


“…53% of projects overrun cost estimates by 189% or more (at a cost of US$59 billion per year in the US alone)”

This research is supported by a US government study on software development projects, which revealed that 60% of projects were behind schedule and 50% were over cost.

So ... were these projects failures?
Estimating: A Brief Note About Project Failure


“…more projects are doomed by poor cost and schedule estimates than by technical, political or team problems.”

Think about that!! The project is doomed because someone did not guess correctly about what might happen over then next 1, 2 or 3 years.

I find it hard to understand how the project is ‘doomed’ by this!

OR did we just build in a major project RISK … ie: possible failure to achieve a given deadline set 3 years ahead.
Estimating: & Uncertainty

1. The future is uncertain – a Universal Law of Nature
2. The further into the future we attempt to look, the greater the Uncertainty
3. When you talk about Process Improvement, are you improving the ability to reduce future Uncertainty?
4. OR are you merely creating more steps, and more checkpoints, that give an appearance of providing certainty?
5. Is Process Improvement really just another name for auditing or enforcing adherence to a nominated process?
6. Are Estimating and Planning really just ‘Ritualistic’ – if you do it, it will succeed ... If it fails, it is because you didn’t do it rigorously enough!
7. Basing estimates on history? You cannot swim in the same river water twice!
Estimating: A Brief Note About Project Failure
Estimating: A Brief Note About Project Failure

- Construction began in 1957
- Scheduled completion date (set by the government was) January 26, 1963
- Original Cost estimate: $7,000,000
- Opened in 1973 - 16 years in construction - 10 years late
- Final Actual Cost: $102 million ... $95,000,000 cost overrun ... 12 times the original budget cost

- 10 years → 16 years ... $7 million → $102 million

- If usual software project success criteria were applied, this was a huge project failure.
Estimating: A Brief Note About Project Failure

- this Modern Expressionist building is one of the most famous, and most photographed, structures of the 20th century.
- In 2007 the Sydney Opera House was one of the finalists in the New Seven Wonders of the World Competition.
- It is an icon of Australia, probably the most recognised symbol of Australia (along with Uluru).
- It is a superb building, a magnificent opera house and theatre complex.
- The building is very much ‘fit for its purpose’.
- The ‘business value’ of the project outcome is enormous.

So, was it a failed project or a highly successful project?
Agile Estimating

So how do you estimate in an agile environment?

Well, much the same way as in a traditional approach, to start with, but then with

- constant adaptation and updating of estimates
- based on empirical observation and measurement of team achievement, productivity and acknowledgement of the changing situation.

But the basic unit of estimating is the ‘Story Point’.

Story Points are the weighting given to each User Story BY THE DEVELOPMENT TEAM that encompasses the perceived complexity, technical difficulty, expected development time of the User Story.
Agile Estimating

*Agile Estimating and Planning* (Robert C. Martin Series) (Paperback) by Mike Cohn (Author)

*User Stories Applied: For Agile Software Development* (The Addison-Wesley Signature Series) by Mike Cohn (Paperback - Mar 11, 2004)
Agile Estimating

- In traditional planning and estimating, which we have seen has usually been highly inaccurate, we are constantly trying to achieve development of ALL REQUIREMENTS in the ESTIMATED COST and the ESTIMATED TIME, by the ESTIMATED DEADLINE.
- And this has usually resulted in project failure.

- In agile development we develop the MOST VALUABLE REQUIREMENTS for the BUDGETTED COST by the STIPULATED DEADLINE.
- At all times the probable set of achieved requirements is known ... by greater transparency and communication.
- At the end-point (no more money or no more time) the LEAST VALUABLE REQUIREMENTS are not achieved.
- An informed decision may then be made if it is even worth pursuing those requirements.
- Constant communication keeps everyone informed.
Agile Estimating

- In any project, a certain level of understanding about the size and complexity of the project must be understood – especially if an up-front ‘estimate’ is demanded, and project approval is based on that estimate.
- Personally, I suggest what are basically JAD sessions.
- Initial JAD sessions are long enough for us to gain sufficient insight into the project to be able to give an indicative estimate.
- BUT ... nobody can foretell the future ... there will always be uncertainty ... the insistence on up-front estimates immediately creates a substantial project risk.
How does Planning Poker work?

The idea behind planning poker is simple. Individual stories are presented for estimation. After a period of discussion, each participant chooses from his own deck the numbered card that represents his estimate of how much work is involved in the story under discussion. All estimates are kept private until each participant has chosen a card. At that time, all estimates are revealed and discussion can begin again.

All team members have the opportunity to justify / support / explain their estimate.

Knowledge sharing is essential.

This is similar to the Delphi method of estimating.
Agile Estimating – WIDEBAND DELPHI APPROACH

- The Delphi approach or Wideband Delphi technique attempts to gather the opinions of a group of experts with the aim of producing an accurate unbiased estimate. It is a structured technique of expert judgement and is essentially a form based technique involving a multistep procedure:
  - Experts are issued the specification and estimation form by the co-ordinator.
  - A group meeting is held to discuss the product and estimation issues.
  - Experts produce an independent estimate.
Agile Estimating – WIDEBAND DELPHI APPROACH

- Estimates are returned indicating the median estimate and the expert’s personal estimate.
- Another group meeting is held to discuss results.
- Experts prepare a revised independent estimate.
- Steps 3-6 are repeated until a consensus is reached by the panel of experts.

In this context, ‘experts’ means the members of the development team, who will be responsible for carrying out the tasks being estimated.

- Estimates can be made in ‘story points’ which are essentially time-free indications of relative effort.
- Empirical evidence will arise about how many ‘story points’ can be achieved by the team in an iteration. This leads to a measure of ‘velocity’ ie: the observed number of story points achieved by the team.
Estimating with User Stories

INVEST in User Stories …

A User Story is a simple statement of a valuable requirement in the proposed system:

Examples:

- _As the Faculty Dean, I want to be able to view the demographics of our students in a visual manner._

- _As a Customer, I want to be able to pay by credit card._

- _As the Payroll Clerk, I want to be able to make direct debits of wages and salaries into employees’ bank accounts._

- _As the Marketing Manager, I want to be able to send bulk emails to our existing customers to advertise special deals._
Estimating with User Stories

User Stories are just that; short stories about what the user or client wants from the system. They are thus very ‘client oriented’.

- The details of these stories are elaborated in a J-I-T manner.
- User Stories are collected at the start, and continually collected, revised, removed and elaborated as the project proceeds.
- The current list of User Stories represents what we know now to be the requirements ... but that will change.
- Each User Story is given a priority, based on various factors including Business Value and potential difficulty.
- Highest priority stories are developed first, therefore the client is always receiving the most ‘valuable’ system components first.
Estimating with User Stories

Calculating Team Velocity

- Team Velocity is the measure of how many User Story Points can be achieved by the Team in a given period of time (an iteration, a Sprint, a Time-Box).
- Team Velocity is a measure based on the empirical evidence of Team achievement.
- It is assumed that the team’s Velocity will change over time, usually for the better, as the Team gains experience, improves productivity, and gets better at estimating story points.
Theorists and Practitioners of Agile Development place emphasis on the human factors in software development.

Team ‘war room’ layout and facilities are seen as an important aspect contributing to team performance and success.

Team co-location in open space is seen to enhance and encourage collaboration, cooperation, communication and knowledge sharing.

Private offices nearby are available for those times when working alone and away from the hubbub of the team activity centre is desirable.

Commons and Caves concept.
Features of Agile Theory, Principle & Practice: Team Co-Location

- A full availability of whiteboards, flip-charts, photocopiers, projection screens, networked analysis and design software, networked decision support software etc. is essential.

- Information ‘radiators’ prominently visible at all times (Kanban).

- Significant reduction in ‘transportation’ and ‘movement’ is achieved (Muda) - one of the ‘wastes’ identified by Lean Product Development activists.
Features of Agile Theory, Principle & Practice: Team Co-Location

- Team members were selected not only for their technical skills and abilities, but also for their 'ability to interact with other team members', and their 'compatibility as team members'.
- The problem of 'knowledge silos', with the often experienced mistrust and communications breakdown, was overcome by directing all team members emotional intensity and commitment towards attainment of the common goal.
- There was a 'deliberate process of total communication'. In team meetings there existed no pecking order and no hidden agenda. (The Team Leader) actively sought contributions from all team members. All ideas were discussed before a final decision was made.

Maani & Cavana (op.cit)
Features of Agile Theory, Principle & Practice:
Step 0: Tools, Standards, Training, Project Preparation

- Hiring competent people,
- Multi-skilled teams, ongoing staff development
- Developing and applying standards:
  - GUI standards
  - Document Standards
  - Code standards
  - Testing and validation standards
  - Process Standards
- Standard Tools
  - Compare Boeing’s 777 development with Airbus’s Airbus380 development.
Features of Agile Theory, Principle & Practice:
Step 0: Tools, Standards, Training, Project Pre-Preparation

- A Diversion: Boeing’s 777 Development:

- A totally new aircraft design
- High level of collaboration with the customer
- Tight development time-line
- Root cause of delays and problems in previous design activities identified and analysed:
  - People not working together,
  - No culture of looking for problems – leave it QA at the end of the line,
  - Lack of prompt and effective communication between developers

- Essentially People Problems
Features of Agile Theory, Principle & Practice:
Step 0: Tools, Standards, Training, Project Pre-Preparation

- Effectively multi-skilled small team workgroups established,

- A culture of “share early and share often” ... knowledge sharing, no secrets, no hidden agendas amongst developers,

- Design / Build teams... members from design, manufacturing, suppliers, customer airlines (pilots, baggage handlers, refuelers ... ),

- Supplier cooperation similar to Toyota’s Supplier Groups ... sharing ideas, goals similar (to improve Toyota’s vehicles),

- “Test early and Fail Fast” ... Testing moved to earliest possible time in the development activity ... continuous testing ... in-line QA not end-of-line QA.

- Supported by sophisticated 3D design tools used by all teams (compare to Airbus A380 electrical harness disaster, and a Mars probe shot that crashed)
Features of Agile Theory, Principle & Practice: Step 0: Tools, Standards, Training, Project Pre-Preparation

- In my view, Step 0 is extremely important.
- The team must be ready to go as a Team.
- To use a sporting analogy, you don’t need a team of champions, but a champion team,
- What makes a champion Team:
  - Trained, compatible, multi-skilled members, able to collaborate, share information, communicate
  - On-going training and knowledge gaining
  - Well defined and agreed standards
  - High level of competence in a useful Tools portfolio (DBMS, IDE, Report Generator, etc. etc.)
  - Visionary Leadership, not Command and Control Management.
  - A well thought out game plane, adaptable based on empirical observation and experience.
Features of Agile Theory, Principle & Practice: Iterative Development / Time-Boxing

- Iterative development is development in short iterations (1, 2 weeks, 30 days)
- Each iteration is fully planned & estimated at the start of the iteration, and produces working, fully tested and ‘production ready’ components,
- Each iteration is a strict time-box. Whatever is unfinished in that iteration is put back on the ‘requirements list’ (called a Product Backlog) and is again analysed, detailed, estimated and considered for inclusion in the next iteration (called a Sprint in one agile method … Scrum)
Features of Agile Theory, Principle & Practice: Iterative Development / Time-Boxing

- Sprints are the time related basis for Just-in-Time requirements determination. User stories are estimated in Story Points, and included in a Sprint, where the details of the User Story are analysed, elaborated, detailed, explained and understood.

- This is not extending the “Scope” of the project ... but extending the detailed understanding of one small part of the overall requirements
Features of Agile Theory, Principle & Practice: Iterative Development / Time-Boxing

Iterative development has a number of advantages:

- There is always a clear deadline a short time in the future
- Constant effort is encouraged in this way,
- The progress of the project is always highly visible to all concerned
- Frequent opportunities to validate and verify requirements,
- Confidence built in the client about the development teams ability to perform and deliver
- Detailed requirements are always ‘fresh’ (Just-in-Time Analysis).
- Agile practice requires continued focus on selected components during a Sprint … no task swapping and disruption of team activity from ‘special requests’ or demands during the Sprint
Features of Agile Theory, Principle & Practice: Iterative Development / Time-Boxing

- Team ‘velocity’ or rythme (also called ‘takt time’) can be established, and final deadlines continually known.
- Use of ‘Burndown Charts’ is now a useful information resource,
- Maximum wastage on failed development is maybe 2 weeks, or 30 days maximum.
- Iterative development assists in the education and learning process. In the traditional approach there is little opportunity to learn about real requirements, beyond the first major Analysis phase.
- The ‘up front’ analysis requirement maximises project risk, project cost.
- In an agile, iterative development activity, requirements of greatest business value are delivered first.
- This means that when the time or cost budget is reached, all that remains are low value requirements.
Features of Agile Theory, Principle & Practice:
Continuous Validation of Requirements (Testing and QA)

- A major part of Agile practice includes:
  - Test Driven Development
  - Test-First development
  - Continuous Testing
  - Continual Integration

- QA is a built-in, in-line activity, not an end-of-line activity.
- QA and Testing is always the responsibility of the developers. There may be QA ‘experts’ in a development team, but it is never intended that they solely involve themselves in QA, or are the only one(s) having that responsibility.
- Iterative development demands and enables continuous validation of requirements, and the production of only fully correct and tested components.
- A human factor here … ability to focus on a small component, rather than a huge amount of code and functionality.
Features of Agile Theory, Principle & Practice: Continuous Validation of Requirements (Testing and QA)

- Tool-based testing
- A number of testing environments available today:
  - FIT
  - xUnit
Features of Agile Theory, Principle & Practice: Project Requirements Backlog and Task Prioritisation

- All requirements are initially stated as User Stories (as discussed before)
- User Stories are a recognition of the facts that
  - Clients are not technically-oriented
  - The system is ultimately to support business activities
  - Clients express their requirements in small stories and anecdotes and descriptions of What we do? And often How we do it? It is the developers responsibility to transform these into technical statements and definitions.
- The total collection of User Stories at any one time defines the total requirements of the system ... as we know it at this point in time.
Features of Agile Theory, Principle & Practice:
Project Requirements Backlog and Task Prioritisation

- The number of User Stories will change over time, including the Client adding more User Stories as they realise and learn more about their requirements, and what is possible. Agile Slogan: “Welcome Change”
- The substance of the User Stories will not change, because we will use a J-I-T detailed analysis approach when the time comes to develop that User Story
- User Stories are estimated in Story Points. A User Story can be compared against another User Story for relative complexity, difficulty, time.
- User Stories are continually prioritised according to the Business Value – as stated by the Client.
- At any time either the probable end date of the project, or the remaining unfinished User Stories, can be stated (by using a Burndown Chart)
Features of Agile Theory, Principle & Practice:
Project Requirements Backlog and Task Prioritisation
Features of Agile Theory, Principle & Practice: Project Requirements Backlog and Task Prioritisation

- Team Velocity

- A metric based on empirical understanding of the team’s productivity in prior Sprints.
- The number of Story Points that are likely to be achieved in the next Sprint.
- Every Sprint, this metric will change, but will tend towards the team’s overall productivity, as estimated in Story Points.
- Burndown can be calculated by

  \[
  \text{Estimated Story Points Remaining} \div \text{Velocity}
  \]
Features of Agile Theory, Principle & Practice: Communication and Collaboration principles and practices

- Team Co-Location
- Information Radiators
- JAD sessions
- Pre-Sprint Planning Meeting
- Daily Standup Meeting
- Post Sprint Retrospective
- Client an integral part of the team
Features of Agile Theory, Principle & Practice: Requirements elicitation and the problem of decaying relevance and validation

Source of Error

- Incomplete Requirements: 56%
- Coding: 7%
- Other: 10%
- Design: 27%

Source: De Marco
Features of Agile Theory, Principle & Practice: Requirements elicitation and the problem of decaying relevance and validation

Requirements Elicitation: OBVIOUSLY Essential!!! BUT ... When and How?

- Why is it necessary to know everything before you can do anything?
- Is “Up Front” the place to find out all?
- Change Control – are we really just guaranteeing that we never deliver a system that people need?
- Is “Acceptance Testing” just prior to Implementation really too little, too late?
Features of Agile Theory, Principle & Practice: Requirements elicitation and the problem of decaying relevance and validation

Requirements Elicitation: OBVIOUSLY Essential!!! BUT ... When and How?

- Upfront analysis and planning maximises project risk!
  - By denying the opportunity to learn,
  - Locking in schedules at a time of least knowledge,
  - Locking in all analysis and design errors,
  - Creating bloated requirements of limited business value
Features of Agile Theory, Principle & Practice: Requirements elicitation and the problem of decaying relevance and validation

- What really is “scope creep”? Change to stated requirements? Extra requirements? Extending the boundary of the system? An outcome of knowing more? Understanding better?
- Should we ignore a ‘learning approach’ where people learn more and understand more about their real requirements?
- Agile Development … Just-In-Time Requirements Elicitation!!!
- Agile and Adaptive Strategy … find out exactly what is needed and develop it immediately – but a small increment – risk reduction.
- Time-boxed development – deliver at a guaranteed time - iteratively.
Features of Agile Theory, Principle & Practice: Requirements elicitation and the problem of decaying relevance and validation

Are We Really Engineering Systems?


- “We need to shift from the old models of software development and maintenance – viz construction and manufacturing - to a new, more resourceful model of software development - software evolution.”

** ie: The Lean development concept that we are not manufacturing or engineering software. Our task is product DEVELOPMENT, NOT product PRODUCTION or MANUFACTURING.
Features of Agile Theory, Principle & Practice: Requirements elicitation and the problem of decaying relevance and validation

- Continuous validation of Requirements.
- Project Requirements Backlog and Task Prioritisation
- Estimating Methods using User Stories, Planning Poker, Story Points
- Iteration Task Selection
- Communication and Collaboration principles and practices
- Releases
- Continuous Integration and Test Drive Development.
- Documentation
- Agile Contracts
Agile Development Published Methods

- EVO www.xs4all.nl/~nrm/EvoPrinc/
- Spiral Model,
- Extreme Programming (www.xprogramming.com/xpmag/whatisxp.htm),
- Scrum (www.controlchaos.com),
- Crystal (alistair.cockburn.us/index.php/Crystal_methodologies_main_foyer)
- Feature Driven Development (FDD) (www.featuredrivendevelopment.com/)
- Dynamic Systems Development Method (DSDM) (www.dsdm.org/)
- Agile Data Modelling Method
Agile Development Published Methods

- EVO [www.xs4all.nl/~nrm/EvoPrinc](http://www.xs4all.nl/~nrm/EvoPrinc),
- Spiral Model,
- Extreme Programming ([www.xprogramming.com/xpmag/whatisxp.htm](http://www.xprogramming.com/xpmag/whatisxp.htm)),
- Scrum ([www.controlchaos.com](http://www.controlchaos.com)),
- Crystal ([alistair.cockburn.us/index.php/Crystal_methodologies_main_foyer](http://alistair.cockburn.us/index.php/Crystal_methodologies_main_foyer))
- Feature Driven Development (FDD) ([www.featuredrivendevelopment.com](http://www.featuredrivendevelopment.com))
- Dynamic Systems Development Method (DSDM) ([www.dsdm.org](http://www.dsdm.org))
- Agile Data Modelling Method
Lean Development
The Toyota Way

- Toyota is renowned as the company that created Lean Manufacturing.
- Toyota is managed according to 14 basic management principles that can be summarised as:
  - Fostering an atmosphere of continuous learning and improvement
  - Satisfying customers (and eliminating waste)
  - Quality first and consistently
  - Grooming leaders from within the organisation
  - Teaching employees to become problem solvers
  - Growing together with suppliers and partners for mutual benefit.
The Toyota Way

14 Principles ... could take hours to narrate. So I have picked and chosen the specific principles I will present.

These can be summarised under 4 major headings:

1. Long-term philosophy
2. The Right Process Will Produce the Right Results
3. Add Value to the Organisation by Developing Your People and Partners
4. Continuously Solving Root Problems Drives Organisational Learning
The Toyota Way

The right process will produce the right results.

- Work process should achieve high value-added, continuous flow. Eliminate wasted time and movement and activities that do not contribute directly towards to final product.
- Continuous flow is the key to a true continuous improvement process and to developing people.
The Toyota Way

Use “pull” systems to avoid overproduction.

- Provide downstream customers in the production system with what they want, when they want it, and the amount they want.
- Material replenishment is initiated by consumption – just-in-time replenishment.
- Minimise WIP and inventories.
- Be responsive to day-to-day demands, not major inventory tracking and maintenance.
The Toyota Way

Build a culture of stopping to fix problems, to get quality right the first time

- Quality for your customer drives your value proposition
- Use all the modern quality assurance methods available
- Develop a visual system to alert team and project leaders that a machine or process needs assistance.
- Build into your organisation support systems to quickly solve problems and put in place counter-measures
- Build into the culture the philosophy of stopping or slowing down to get quality right the first time to enhance productivity in the long-term.
The Toyota Way

Standardised Tasks are the foundation for continuous improvement and employee empowerment.

- Use stable, repeatable methods to maintain predictability, regular timing and regular output of your process.
- Capture the accumulated learning about a process up to a point in time by standardising today’s best practice.
- Allow creative expression to improve on the standard, then incorporate it into the new standard.

- Use visual controls so that no problems are hidden.
- Use only reliable, thoroughly tested technology that serves your people and process.
The Toyota Way

- Add Value to the Organisation by Developing Your People and Partners
  - Grow leaders who thoroughly understand the work, live the philosophy and teach it to others
    - Eg: a good leader must understand the daily work in great detail so he or she can be the best teacher of the company philosophy.
  - Develop exceptional people and teams who follow your company’s philosophy.
  - Respect your extended network of partners and suppliers by challenging them and helping them improve.
The Toyota Way

- Continuously Solving Root Problems Drives Organisational Learning

  - Go and see for yourself to thoroughly understand the situation.
  
  - Make decisions slowly by consensus, thoroughly considering all options; implement decisions rapidly.
  
  - Become a learning organisation through relentless reflection and continuous improvement.
The Toyota Production System (TPS)

The Other Dimension of The Toyota Way
The Toyota Production System

TPS is Toyota’s approach to manufacturing
It is the basis for “Lean Manufacturing”

Lean Manufacturing is a 5 step process

1. Define Customer Value
2. Define the Value Stream
3. Make the manufacturing process “flow”
4. “Pull” based manufacturing from the Customer back
5. Strive for Excellence
To be a “lean manufacturer” requires a way of thinking that focuses on:

- Make the product flow through value-added processes without interruption.
- Have a “pull” system that cascades back from customer demand by replenishing only what the new operation takes away in short intervals.
- Develop a culture where everyone is striving for continuous improvement.
- Toyota’s initial focus in the TPS was eliminating waste and material from every step of the process.
The Toyota Production System

Some counter-intuitive truths within the TPS:

- Often the best thing that you can do is idle a machine. This avoids overproduction, the creation of inventories, which is seen as a fundamental waste.
- Building up an inventory in order to level out production, is often best.
- It may not be top priority to keep your workers busy making parts as fast as possible. Otherwise, this leads to overproduction, which is a form of waste.
- It is best to selectively use information technology and often better to use manual processes, even when automation may appear to be able to reduce labour costs.
Lean Software Development

Applying The Toyota Production System Principles to Software Projects
Lean Software Development


The Chapters in this book illustrate the thinking, and to a great extent the influence of the TPS:

- Chapter 1: Eliminate Waste
- Chapter 2: Amplify Learning
- Chapter 3: Decide as Late as Possible
- Chapter 4: Deliver as Fast as Possible
- Chapter 5: Empower the Team
- Chapter 6: Build Integrity In
- Chapter 7: See the Whole
Shigeo Shino, one of the masterminds of the TPS identified seven types of waste:

<table>
<thead>
<tr>
<th>The 7 Wastes of Manufacturing</th>
<th>The 7 Wastes of Software Development</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inventory</td>
<td>Partially Done Work</td>
</tr>
<tr>
<td>Extra Processing</td>
<td>Extra Processes</td>
</tr>
<tr>
<td>Overproduction</td>
<td>Extra Features</td>
</tr>
<tr>
<td>Transportation</td>
<td>Task Switching</td>
</tr>
<tr>
<td>Waiting</td>
<td>Waiting</td>
</tr>
<tr>
<td>Motion</td>
<td>Motion</td>
</tr>
<tr>
<td>Defects</td>
<td>Defects</td>
</tr>
</tbody>
</table>
Lean Software Development

Waste # 1: Partially Done Work

- Code that is not tested, integrated and delivered to the customer has a tendency to become obsolete.
- Code should be required, completed, tested, integrated with the rest of the system and delivered to the customer.
- A substantial investment in partially completed code can result in a cost write-off.
- Minimising the inventory of partially completed software is a risk reduction strategy as well as a waste reduction strategy.
- Development strategies such as formal iterations, “sprints” (from SCRUM), User stories (from Extreme Programming), with fully created deliverables at the end of a short iteration manifest this strategy.
Lean Software Development

Waste # 2: Extra Processes

- Documentation, Documentation, Documentation,!!!!
- The traditional development approaches tend to emphasise documentation over production of code.
- Quote: A VP – Business Systems, of a major Singapore company (Capitaland) “Roy, why do IT people always want to produce large documents?”
- Three anecdotes here ...
  - FAQ’s in IT project unit
  - ACIS presentation by an academic
  - Discussion with a project supervisor

Question: What illustrates satisfactory project progress more ... a large set of documentation, or a large set of delivered code?
Another Question: What would you prefer ... delivered code without documentation, or delivered documentation without code?
Lean Software Development

Waste # 3: Extra Features
- Research has shown that up to 60% or so of features in system are never used.
- Providing features that are not requested, just in case, is a waste.

Waste #4: Task Switching
- Multi-tasking (as in attempts at parallel tasking) wastes time.
- “Switch off” and “Switch On” times are a waste.
- Single tasks are stated in “sprints” (in SCRUM) or in focused iterations.
Lean Software Development

Waste # 5: Waiting
- Delays in starting a project,
- Delays in staffing the project,
- Delays due to excessive documentation requirements,
- Delays in finding out information from the client
- Delays in having reviews and approvals and testing
- Delays in deployment
- How long does it take to get a decision?

- A fundamental lean development principle is to delay design decisions until the last possible moment, so that you have the best information available to you, and then quickly develop the code and deliver it.

- Compare to the traditional approach where design decisions are apparently made “up front” and may be obsolete and out-dated by the time the developers come to implement it.
Lean Software Development

Waste # 6: Motion

- How far away are the decision makers?
- Where does a developer have to move to, to have a question answered?
- Are people on hand, nearby and available to help with technical matters?
- Is there a customer representative on-hand and accessible to quickly answer questions about features?
- Does a developer have to physically move any distance, or to another building, to have questions answered, or test results delivered?
- Lean / Agile development practices recommend a team environment in a single room, with close client participation as part of the team.
- Movement, handoffs of artefacts, handovers cause waste.
Lean Software Development

Waste # 7: Defects

- The amount of waste caused by a defect is a function of the defect impact and the time it goes undetected.
- On the Toyota line, the whole production line is closed down as soon as a defect is seen, demanding immediate rectification.
- In software development, requiring a fully developed, fully tested, defect free component as the outcome of a “sprint” or iteration is the equivalent.
- Students and inexperienced developers tend to leave the “debris field” behind them, leading to the wreck of the system.
Lean Software Development

Value Stream Mapping

- A valuable project management strategy is to analyse the value stream ... those things that directly add value to the product ... and eliminate those things that do not.
- This discovers the waste in the development process.
The Agile Manifesto

*We follow these principles:*

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.
Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.
Continuous attention to technical excellence and good design enhances agility.

Simplicity—the art of maximizing the amount of work not done—is essential.

The best architectures, requirements and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.
Importance of this research into Agile Development and Lean Development Principles

- Simply put ...
  - Built-In Quality,
  - Useful and Useable Systems delivered as a usual outcome,
  - Savings of literally billions of dollars
Agile Methods & CMMI

1. Fundamentally I cannot see any real tension between CMMI and Agile
2. Given that CMMI is a method of assessing the effectiveness of a development approach
3. It is not a development approach itself
4. It is not a process compliance audit method
The Future for Agile Methods in Thailand

1. I would like to see the creation of a community of “agileists”, drawn from industry and academia
2. Research into Agile Methods effectiveness
3. Research into Agile Methods & CMMI
4. Funds made available for small scholarships for Masters and other HDR students for research into agile methods.
5. Organisations opening up to allow action research by HDR students and other researchers

6. Why? Because I think a greater understanding of agile methods, and greater adoption in industry, can have a major economic impact on IT / IS system development in Thailand.
Questions & Discussion

Thank you for your attention and interest
My email address at Naresuan University is roym@nu.ac.th