Agile SDLC's

- Speed up or bypass one or more life cycle phases
- Usually less formal and reduced scope
- Used for time-critical applications
- Used in organizations that employ disciplined methods

Some Agile Methods

- Adaptive Software Development (ASD)
- Feature Driven Development (FDD)
- Dynamic Software Development Method (DSDM)
- Rapid Application Development (RAD)
- Scrum
- Extreme Programming (XP)
- Rational Unify Process (RUP)

Extreme Programming - XP

For small-to-medium-sized teams developing software with vague or rapidly changing requirements

Coding is the key activity throughout a software project

- Communication among teammates is done with code
- Life cycle and behavior of complex objects defined in test cases – again in code

XP Practices (1-6)

1. Planning game – determine scope of the next release by combining business priorities and technical estimates
2. Small releases – put a simple system into production, then release new versions in very short cycle
3. Metaphor – all development is guided by a simple shared story of how the whole system works
4. Simple design – system is designed as simply as possible (extra complexity removed as soon as found)
5. Testing – programmers continuously write unit tests; customers write tests for features
6. Refactoring – programmers continuously restructure the system without changing its behavior to remove duplication and simplify

XP Practices (7 – 12)

7. Pair-programming -- all production code is written with two programmers at one machine
8. Collective ownership – anyone can change any code anywhere in the system at any time.
9. Continuous integration – integrate and build the system many times a day – every time a task is completed.
10. 40-hour week – work no more than 40 hours a week as a rule
11. On-site customer – a user is on the team and available full-time to answer questions
12. Coding standards – programmers write all code in accordance with rules emphasizing communication through the code

XP is “extreme” because

Commonsense practices taken to extreme levels

- If code reviews are good, review code all the time (pair programming)
- If testing is good, everybody will test all the time
- If simplicity is good, keep the system in the simplest design that supports its current functionality. (simplest thing that works)
- If design is good, everybody will design daily (refactoring)
- If architecture is important, everybody will work at defining and refining the architecture (metaphor)
- If integration testing is important, build and integrate test several times a day (continuous integration)
- If short iterations are good, make iterations really, really short (hours rather than weeks)

XP References

Online references to XP at

- http://www.extremeprogramming.org/
- http://c2.com/cgi/wiki?ExtremeProgrammingRoadmap
- http://www.xprogramming.com/

Feature Driven Design (FDD)

Five FDD process activities

1. Develop an overall model – Produce class and sequence diagrams from chief architect meeting with domain experts and developers.
2. Build a features list – Identify all the features that support requirements. The features are functionally decomposed into Business Activities steps within Subject Areas.
   - Features are functions that can be developed in two weeks and expressed in client terms with the template: <action> <result> <object>
   - i.e. Calculate the total of a sale
3. Plan by feature -- the development staff plans the development sequence of features
4. Design by feature -- the team produces sequence diagrams for the selected features
5. Build by feature – the team writes and tests the code

http://www.nebulon.com/articles/index.html

Dynamic Systems Development Method (DSDM)

Applies a framework for RAD and short time frames

Paradigm is the 80/20 rule

– majority of the requirements can be delivered in a relatively short amount of time.

DSDM Principles

1. Active user involvement imperative (Ambassador users)
2. DSDM teams empowered to make decisions
3. Focus on frequent product delivery
4. Product acceptance is fitness for business purpose
5. Iterative and incremental development - to converge on a solution
6. Requirements initially agreed at a high level
7. All changes made during development are reversible
8. Testing is integrated throughout the life cycle
9. Collaborative and co-operative approach among all stakeholders essential

DSDM Lifecycle

- Feasibility study
- Business study – prioritized requirements
- Functional model iteration
  - risk analysis
  - Time-box plan
- Design and build iteration
- Implementation

Adaptive SDLC

Combines RAD with software engineering best practices

- Project initiation
- Adaptive cycle planning
- Concurrent component engineering
- Quality review
- Final QA and release

Adaptive Steps

1. Project initialization – determine intent of project
2. Determine the project time-box (estimation duration of the project)
3. Determine the optimal number of cycles and the time-box for each
4. Write an objective statement for each cycle
5. Assign primary components to each cycle
6. Develop a project task list
7. Review the success of a cycle
8. Plan the next cycle

Tailored SDLC Models

- Any one model does not fit all projects
- If there is nothing that fits a particular project, pick a model that comes close and modify it for your needs.
- Project should consider risk but complete spiral too much – start with spiral & pare it done
- Project delivered in increments but there are serious reliability issues – combine incremental model with the V-shaped model
- Each team must pick or customize a SDLC model to fit its project

Agile Web references

DePaul web site has links to many Agile references

http://se.cs.depaul.edu/ise/agile.htm

Mobile Voucher
Solusi e-Marketing Berbasis Mobile

Dalam dunia yang makin kompetitif ini penggunaan model-model pemasaran baru dipercaya mampu mendorong minat orang untuk membeli produk yang ditawarkan. Penggunaan *voucher* belanja telah menjadi salah satu cara bagi perusahaan untuk mempromosikan produk-produknya dan menarik minat calon pembeli untuk membeli. Salah satu kendala yang dihadapi dalam penggunaan voucher belanja ini adalah pada distribusi *voucher* kepada calon pembeli dan pengelolaan *voucher* oleh calon pembeli tersebut. *Mobile voucher* mencoba menjawab permasalahan tersebut dengan menghadirkan *voucher* belanja elektronik yang dapat didistribusikan melalui jaringan telepon seluler dan disimpan oleh calon pembeli sebagai teks SMS.

- Sistem *mobile voucher* terdiri dari 2 bagian, yaitu *voucher management System* dan *customer wallet*. *Voucher management System* adalah sistem komputer yang mengelola *voucher*, terdiri dari sebuah server yang terhubung dengan SMS gateway, bertugas untuk mendistribusikan *voucher* belanja. *Customer Wallet* adalah aplikasi Java SIM Card yang berfungsi untuk memvalidasi dan memverifikasi *voucher* yang dikirimkan melalui SMS oleh *Voucher Management System* berdasarkan kode keamanan tertentu.
- Proyek ini bertujuan untuk membuat aplikasi *Customer Wallet*, sebagai bagian dari keseluruhan aplikasi *Mobile Voucher*. Produk yang dihasilkan adalah perangkat lunak *Customer Walet* berbasis *Java Smart Card API*.
- Proyek ini dilaksanakan dengan tahapan-tahapan: pendefinisian kebutuhan, desain, koding, dan pengujian, dengan jangka waktu pengerjaan selama 2 bulan dan biaya sebesar Rp. 99.999.999.

- ”Bagaimanakah perangkat mobile seperti handphone dapat digunakan sebagai solusi *voucher* elektronik?” Lebih spesifik lagi, ”Bagaimana membuat aplikasi di dalam handphone yang dapat mengelola voucher elektronik?”
- Model Pengembangan Perancangan Perangkat Lunak apa yang baik digunakan (menurut pendapat masing-masing kelompok), buat langkah-langkah secara umum disesuaikan dengan metode yg dipilih.

Quality – the degree to which the software satisfies stated and implied requirements

- Absence of system crashes
- Correspondence between the software and the users' expectations
- Performance to specified requirements

Quality must be controlled because it lowers production speed, increases maintenance costs and can adversely affect business

Quality Assurance Plan

- The plan for quality assurance activities should be in writing
- Decide if a separate group should perform the quality assurance activities
- Some elements that should be considered by the plan are: defect tracking, unit testing, source-code tracking, technical reviews, integration testing and system testing.

Quality Assurance Plan

- Defect tracing – keeps track of each defect found, its source, when it was detected, when it was resolved, how it was resolved, etc
- Unit testing – each individual module is tested
- Source code tracing – step through source code line by line
- Technical reviews – completed work is reviewed by peers
- Integration testing -- exercise new code in combination with code that already has been integrated
- System testing – execution of the software for the purpose of finding defects.