# *Software Development Methodologies*

Jonathan Hoyle

Eastman Kodak

Thursday, June 2, 2005

# *Overview*

- Predictive Methodologies
  - Waterfall
  - Other Predictive Methodologies
- Agile Methodologies
  - Extreme Programming (XP)
  - Other Agile Methodologies
  - Pros & Cons
- World Trade Center Project

# *Predictive Methodologies*

- A reaction to “Cowboy Programming”
- A planned development life cycle
- Using practices which favor a well-defined outcome at each stage
- Plan-driven
- Tends to be Document-based
- Should include all phases of product development: Engineering, QA, Documentation, etc.
- Come to be identified with *Waterfall* methodology

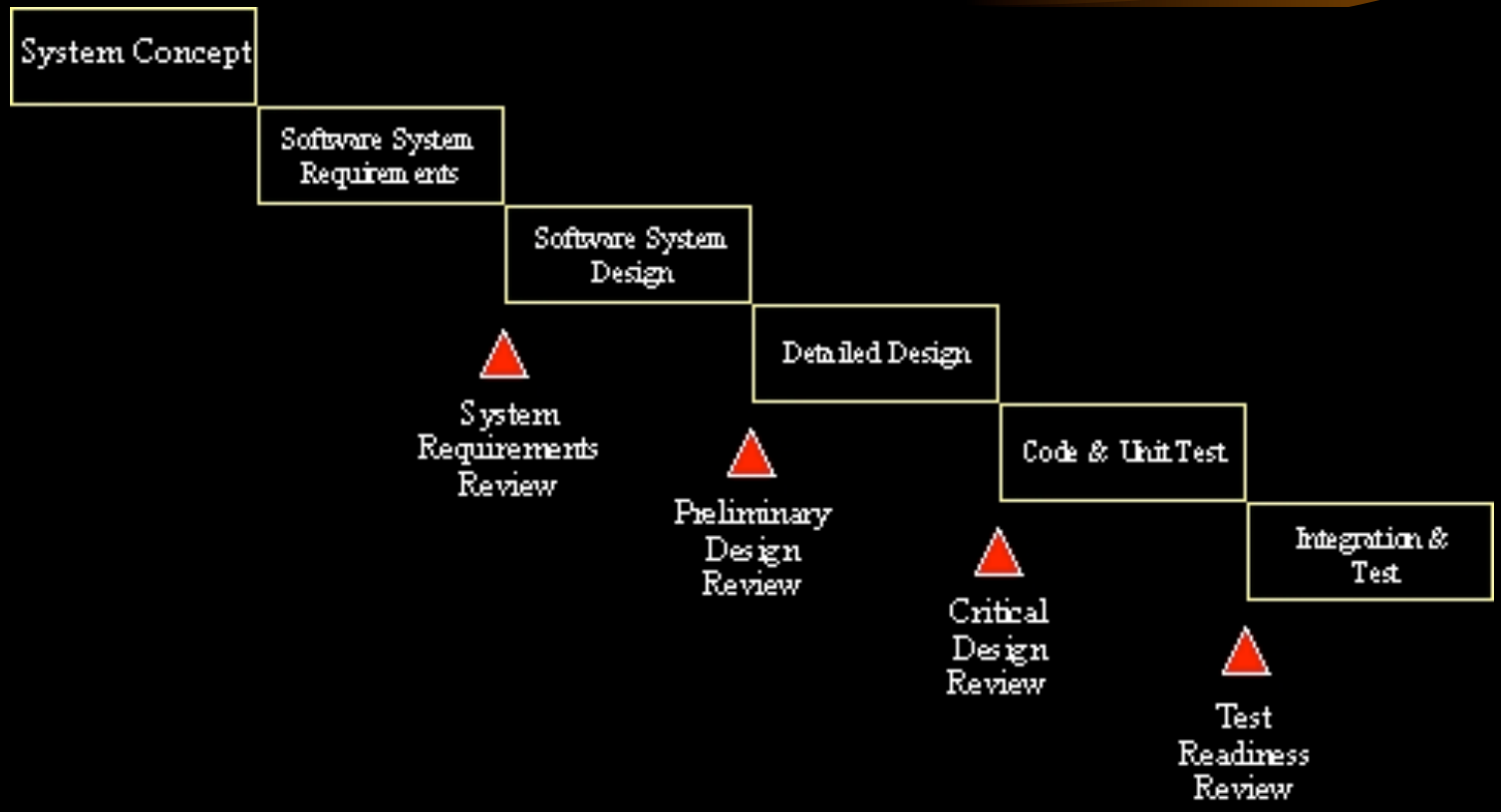
# *Waterfall Methodology*

- First proposed by Winston W. Royce in 1970
- Traditional approach still most dominant
- Can be iterative (Royce encouraged this)
- Motivation:
  - institute a controlled process
  - incorporate risk management
  - incorporate proper design principles
  - estimation & progress tracking
- More robustly defined by CMM / CMMI

# *Waterfall Process*

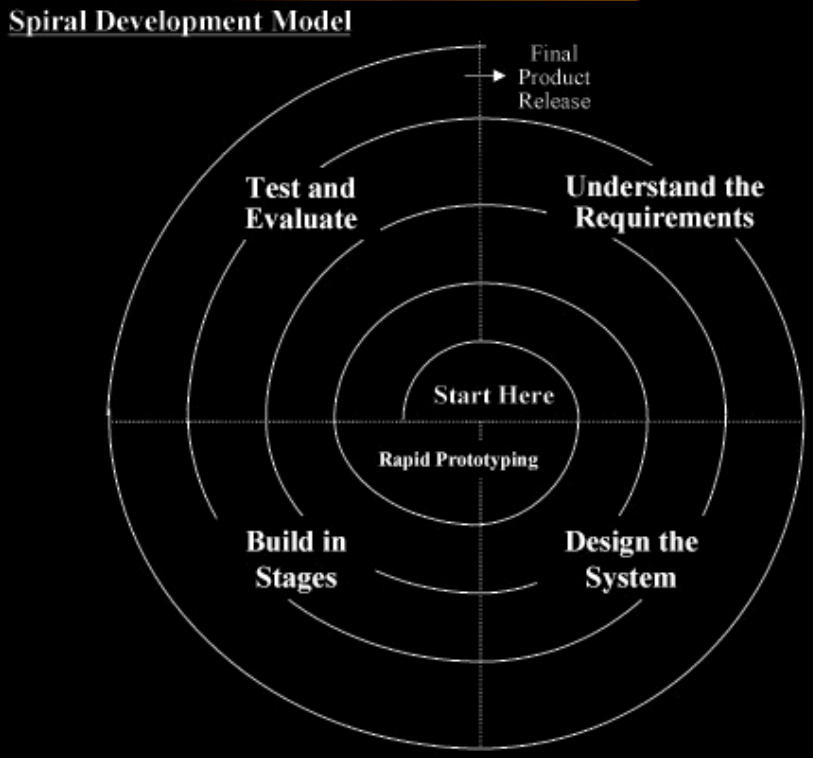
- Orderly sequence of distinct phases or stages
- Each phase is well-defined, following:
  - Analysis
  - (Specify)
  - Design
  - Coding
  - Testing
- Tasks can be pre-planned
- Problems analyzed and resolved well in advance
- Milestones and deliverables clearly defined

# *Waterfall Model Flow*



# *Other Predictive Methodologies*

- Spiral Methodology
  (iterative Waterfall)




- SSADM
- Clean Room Design
- Rational Unified Process (RUP) by Rational Software

# *Agile Methodologies*

- Welcomes change in requirements, even late into the development cycle
- Reaction to “thick” or “heavyweight” methodologies
- The motivation is to create a process by which software development can handle change without sacrificing good programming principles
- Create software iteratively, adding features as you go, rather than wait until the end for a working app
- Voice of customer part of the process

# *Agile Methodology Manifesto*

We are uncovering better ways of developing software by doing it and helping others do it.

Through this work we have come to value:

**Individuals and interactions** over processes and tools

**Working software** over comprehensive documentation

**Customer collaboration** over contract negotiations

**Responding to change** over following a plan

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

# *Extreme Programming (XP)*

- The most well known of Agile methodologies
- Created by Kent Beck in 1996
- A growing community of software developers
- Best known for contributions in these areas:
  - Testing (Test first philosophy)
  - Design (Simplicity & Refactoring)
  - Coding (Paired programming)
  - Iterations (User stories, small fast releases)

# *XP: Testing*

- Test First Philosophy
  - Before implementing a feature or fixing a bug, a unit test is written **first**
  - Verify the test fails (since no code is written)
  - Now write the code to get the test to pass
- All unit tests must pass before checking in
- If any code causes unit tests to fail, those changes are backed out via source control
- All code must be associated with some unit test

# *XP: Design & Coding*

- All production code is *Pair Programmed*
- Pairs swapped on a regular basis
- No code ownership
- All code must conform to agreed upon standards
- Design: *"The simplest thing that can possibly work."*
- Refactor whenever possible (continuous design)
- Little to no comments in the code
- No functionality is added early
- Optimization saved for last

# *XP: Iterations*

- Frequent small releases (iterations)
- Features broken down as “user stories”
- Stories get estimated by engineers and customer determines which stories go in that iteration
- Stories that are too big must be broken down
- Daily “stand up” meetings with all hands
- Retrospective after each iteration (no management)
- Fix XP when it breaks (change what doesn’t work)
- NO  OVERTIME

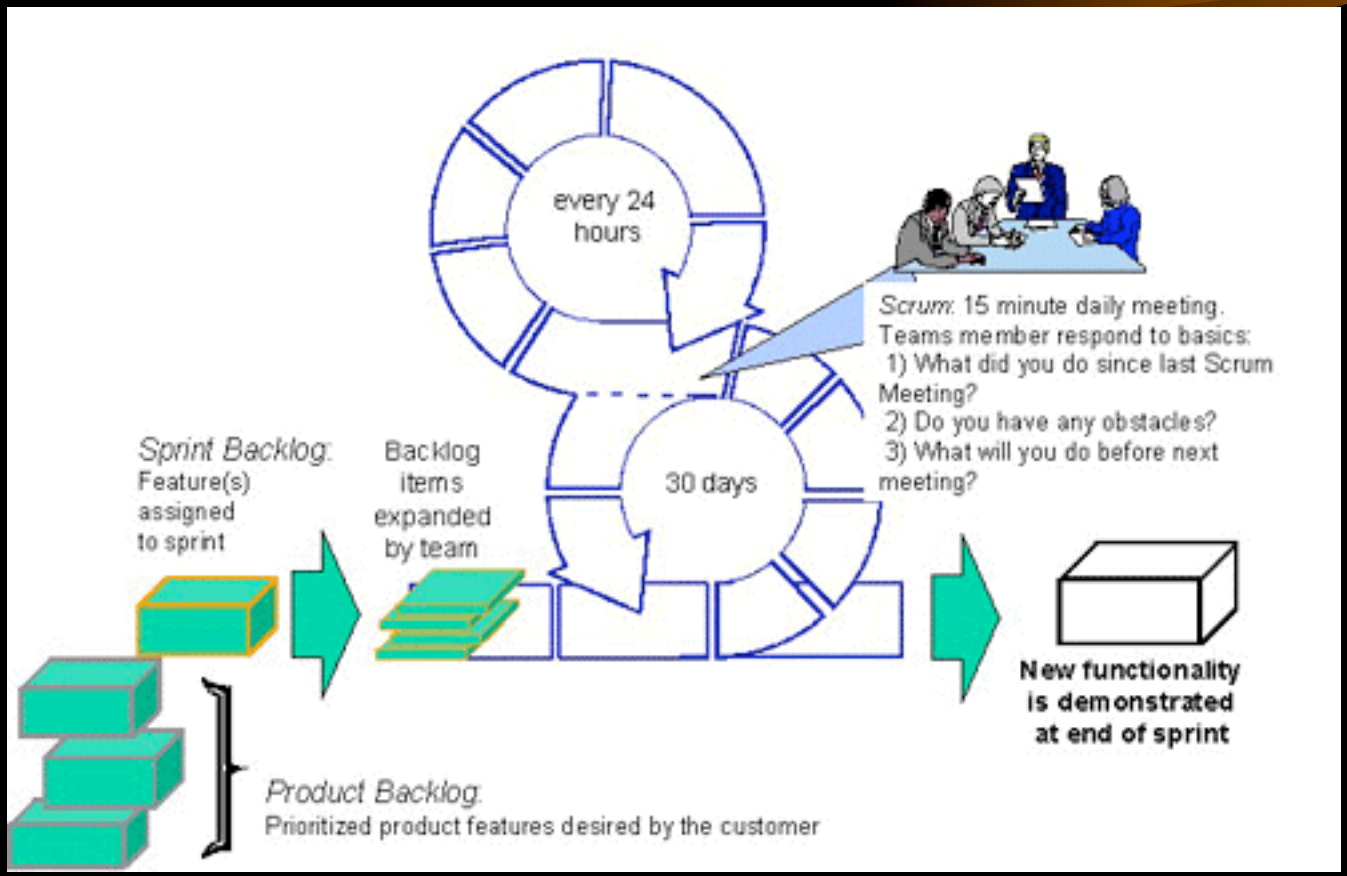
# *XP: World Trade Center Project*

- When Gene Codes Corp. was contracted to create the forensic DNA identification software to identify the victims of the World Trade Center, XP was the methodology chosen to create this software.
- One of the most successful XP projects
- My personal experience with this is explained later.

# *Other Agile Methodologies: Scrum*

- A wrapper around current processes
- Management prioritizes feature list
- 30 day “sprint” for development & QA
- At the end of each sprint, engineers demo the software showing changes since previous sprint
- A releasable version with each monthly cycle
- 15 minute daily team meeting:
  - What did you do since last time? Any obstacles? etc.
  - Discuss current backlog for this sprint
  - Load balance work amongst team, if needs be

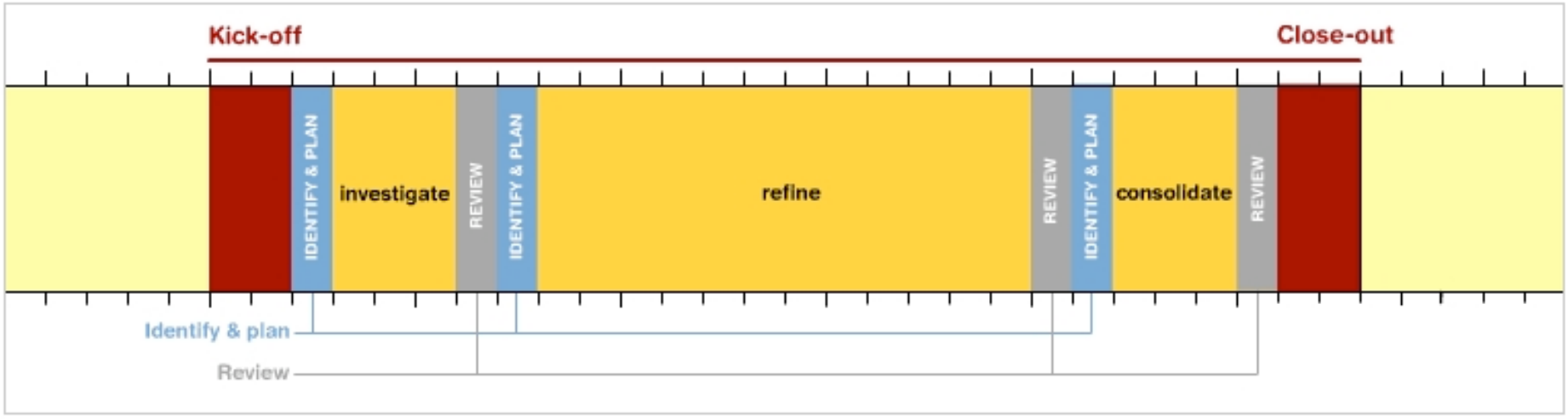
# Scrum


every 24 hours
30 days
Scrum: 15 minute daily meeting.
Teams member respond to basics:
1) What did you do since last Scrum Meeting?
2) Do you have any obstacles?
3) What will you do before next meeting?
Sprint Backlog:
Feature(s) assigned to sprint
Backlog items expanded by team
New functionality is demonstrated at end of sprint
Product Backlog:
Prioritized product features desired by the customer

# *Other Agile Methodologies: DSDM*

- Management delivers prioritized list of requirements
- Prototype is built using MoSCoW principle:
  - **M**UST have this
  - **S**HOULD have this, if at all possible
  - **C**OULD have this, if it doesn't affect anything else
  - **W**ON'T have this, but will in the future
- Each "refinement" is strictly time-boxed (dropping lowest priorities as time or money runs out)
- At the end of a time-box, management reviews, modifies requirements, creates new priorities, etc.
- Lather, rinse and repeat

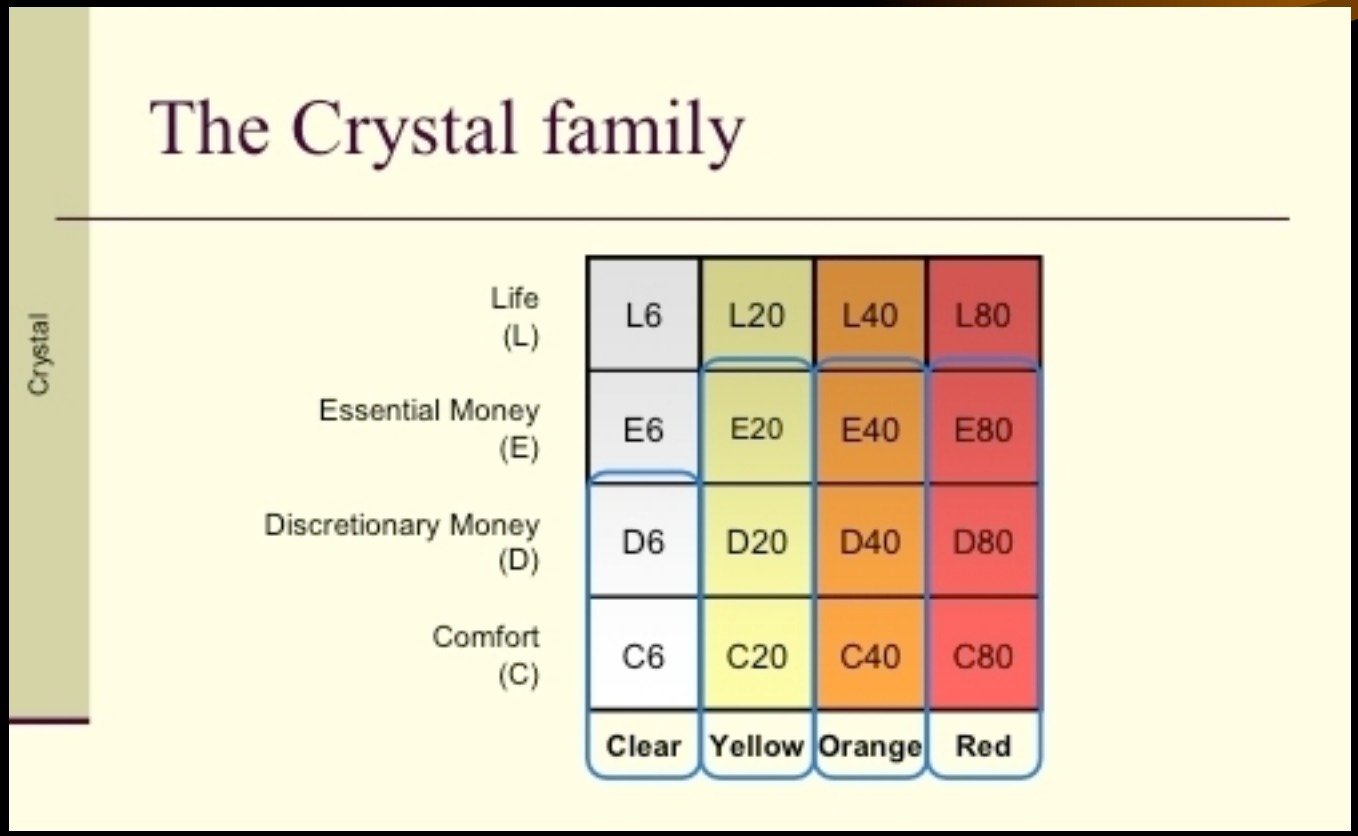
# *DSDM*


Kick-off
Close-out
IDENTIFY & PLAN
investigate
REVIEW
IDENTIFY & PLAN
refine
REVIEW
IDENTIFY & PLAN
consolidate
REVIEW
Identify & plan
Review

# *Other Agile Methodologies: Crystal*

- Differing levels of agility (opacity): Crystal Clear, Crystal Yellow, Crystal Orange, Crystal Red
- Larger teams use less agility
- Software delivered incrementally (2-3 months)
- Automated regression testing
- Direct user involvement
- Team must hold pre- and post-Increment workshops
- Mid-increment evaluation for course-correction

# *Crystal*


Crystal
The Crystal family
Life (L)
Essential Money (E)
Discretionary Money (D)
Comfort (C)
L6
L20
L40
L80
E6
E20
E40
E80
D6
D20
D40
D80
C6
C20
C40
C80
Clear
Yellow
Orange
Red

# *Agile Pros & Cons*

- **Works well with:**
  - Volatile changes in requirements
  - Iterative releases
  - Consistent small team of developers
- **Doesn't work well with:**
  - Large or distributed development teams
  - Multiple project environment
  - Inflexible corporate culture
  - Heavily pre-designed project
  - A pre-scheduled timeline covering a lengthy period