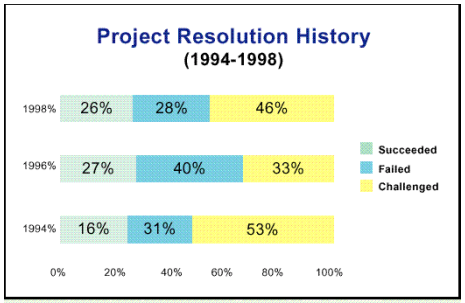
Agile Software Engineering

Frank Maurer
University of Calgary
Computer Science
E-Business Engineering Group
maurer@cpsc.ucalgary.ca
http://sern.ucalgary.ca/~maurer

http://www.standishgroup.com/sample_research/chaos_1994_1.php
http://www.standishgroup.com/sample_research/PDFpages/chaos1998.pdf

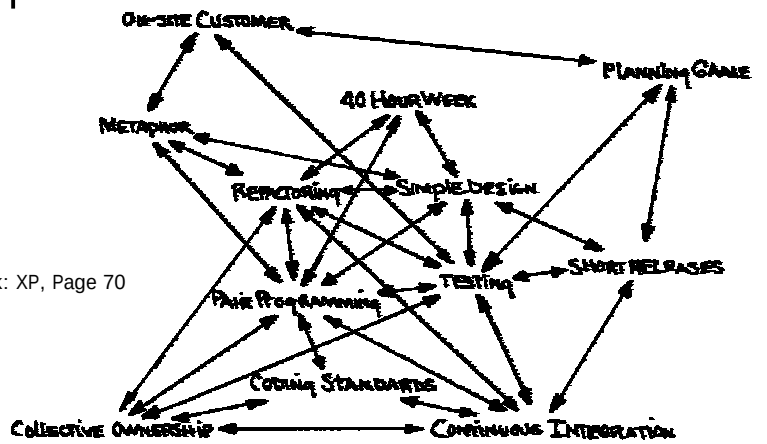
Why agile methods

- CHAOS Report, Standish Group, 1994
  - 31.1%: projects canceled before completion
  - 52.7% of projects will cost 189% of their original estimates
  - 16.2%: projects on-time and on-budget (larger companies: 9%)
  - Projects completed have only approximately 42% of the originally-proposed features and functions.
  - Smaller companies: 78.4% projects will get deployed with at least 74.2%



Project success rates are rising. As shown in the resolution of the 23,000 applications projects in large, medium, and small cross-industry, U.S. companies tested by The Standish Group since 1994.

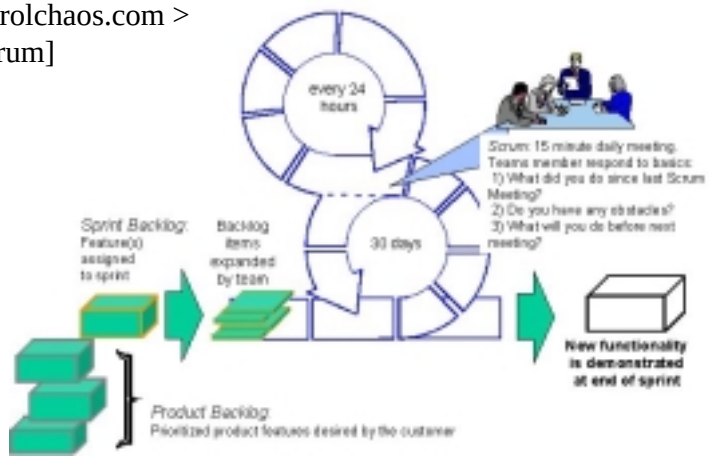
Extreme Programming



From Beck: XP, Page 70

Scrum
(Sutherland, Schwaber and Beedle)

[From controlchaos.com > What is Scrum]



http://agilemanifesto.org/

Manifesto for Agile Software Development

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 negotiation
- 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.

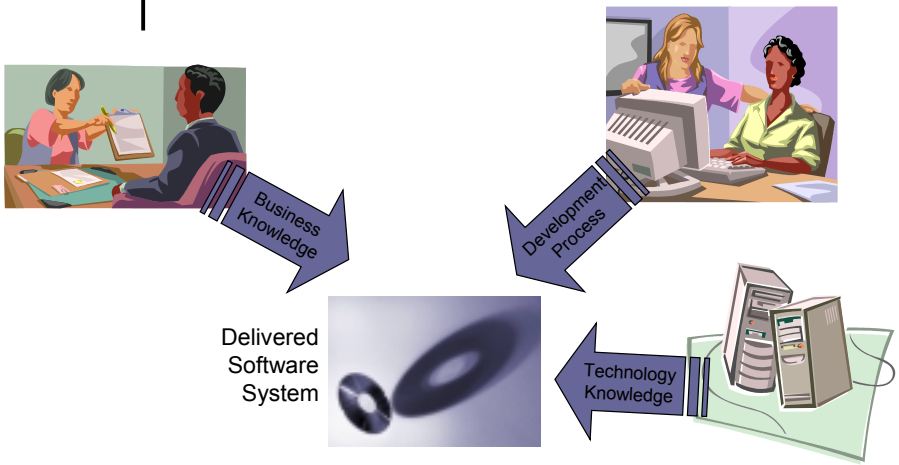
Kent Beck - Mike Beedle - Arie van Bennekum - Alistair Cockburn - Ward Cunningham - Martin Fowler - James Grenning - Jim Highsmith - Andrew Hunt - Ron Jeffries - Jon Kern - Brian Marick - Robert C. Martin - Steve Mellor - Ken Schwaber - Jeff Sutherland - Dave Thomas

The Business Case for Agile Methods

Agile versus tayloristic methods

- Agile methods
  - Human-centric
  - Tacit knowledge sharing
  - Code-centric
  - Replace documentation by face-to-face communication
  - Generalists
  - Plan and correct
  - Customer-focused
- Tayloristic methods (plan-driven, traditional, heavyweight)
  - Process-centric
  - Explicit knowledge
  - Documentation-centric
  - Role specialization
  - Plan and control
  - Contract-focused

Knowledge sharing needed



Communication chains result in knowledge loss

Customer → Analyst → Architect → Designer → Chief programmer → Coder

10 % communication error: 59 % of information gets to coder
5 % communication error: 77 % of information gets to coder

Customer → Developer

10 % communication error: 90 % of information gets to developer
5 % communication error: 95 % of information gets to developer

Business contracts

- Fixed scope/fixed price contracts
  - Trust by contract
  - Contract requires documentation → not lightweight
  - Opposing sides of table
  - Overestimating effort
  - High costs for change requests
- Time and expenses: Fixed cost and fixed programmer hours
  - Trust by feedback and involvement
  - Collaborative environment
  - Changes easy
  - Issues:
    - No time limit on project
    - No guaranteed functionality

Does it really work?

Productivity of XP

- Industrial case study
- 16 months: 9+2+5
- Stable setting (same team, same customer, same project)
- Document-oriented/ad-hoc
  → partial adoption of XP

Agile Software Engineering - © 2003 EBE Group

Productivity

#NLOC / billable hours	+66.3%
#new methods / billable hours	+302.1%
#new classes / billable hours	+282.6%

Agile Software Engineering - © 2003 EBE Group

Productivity (cont.)

#bugs fixed / billable hours	+46.9%
#new features / billable hours	-23.9%
#bugs+features / billable hours	-4.96%

Agile Software Engineering - © 2003 EBE Group

LOC metrics changes

	JLOC	HLOC	NLOC
Ave Pre-XP	4177.22	3267.67	7444.89
Average XP	10608.00	2194.40	12802.40
% Change	**153.9%**	**-32.8%**	**72.0%**

Features/hour: -4.96%

Agile Software Engineering - © 2003 EBE Group

Interpretation

$\frac{\#NLOC}{\#bugs+\#features}$	+75%
$\frac{\#methods}{\#bugs+\#features}$	+323%
$\frac{\#classes}{\#bugs+\#features}$	+302%

Agile Software Engineering - © 2003 EBE Group

Issues with the study

- Billable hours – how accurate
- Why is NLOC going up?
- 16 data points from 1 company
- How much of the changes can be attributed to agile practices?
- Still pretty ad-hoc

Agile Software Engineering - © 2003 EBE Group

The Perceptions Study. Logistics

- Goal: To find out ~~developer~~ student perceptions on agile methods
- Total numbers of respondents: **102**
  - out of 221 invited, 46% response ratio
  - voluntarily and anonymously
    - hypothesis: voluntarism might lead to observation of the extreme ends of the spectrum
- Over 2 years (4 academic semesters)

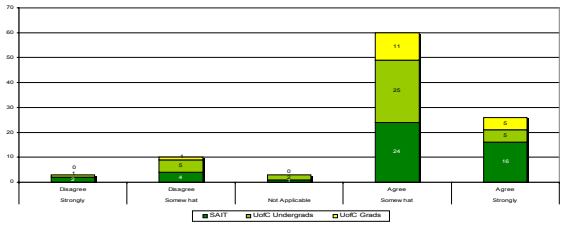
Agile Software Engineering - © 2003 EBE Group

Questions Asked

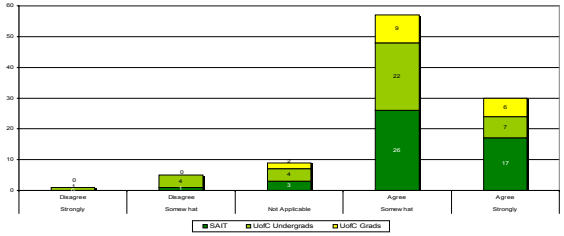
- Did the students enjoy agile practices?
- What worked for them?
- What problems did they encounter?
- Whether they would use agile practices in the future (if allowed)?
- What were their impressions of individual practices?
- How did XP improve their learning?

Agile Software Engineering - © 2003 EBE Group

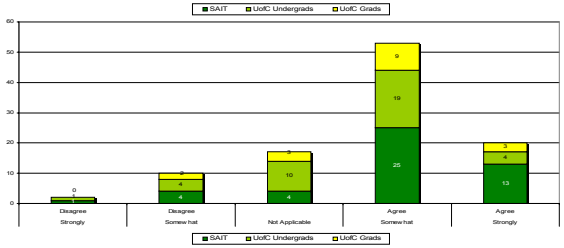
Using XP improves the productivity of small teams



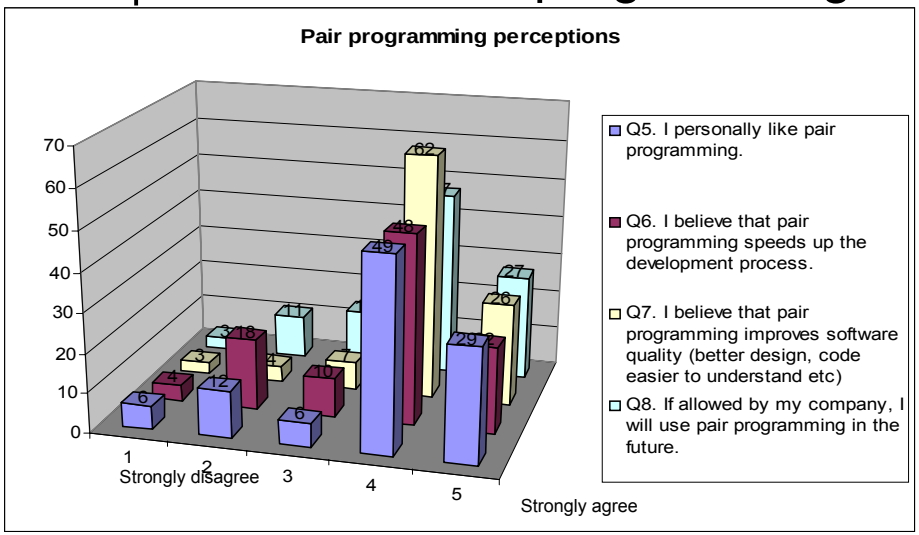
Using XP improves the quality of code



I would recommend to my company to use XP

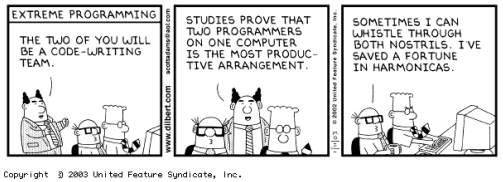


Perceptions on pair programming



Cost of pair programming

- Williams, Laurie, Kessler, Robert R., Cunningham, Ward, and Jeffries, Ron, *Strengthening the Case for Pair-Programming*, *IEEE Software,* July/Aug 2000 .
  - University study with 41 students
  - Higher quality code
    - Test cases passed individuals: 73.4%-78.1%
    - Test cases passed pairs: 86.4%-94.4%
  - Pairs completed assignments 40-50% faster (average 15% higher costs)
  - Pair programming preferred by students (85%)



Copyright © 2003 United Feature Syndicate, Inc.

The Verdict

- The perceptions of XP practices are overwhelmingly positive
  - holds for XP in general and for individual practices in particular
  - holds across all levels of students

ENTHUSIASM

Questioning Ourselves

- Are the results optimistic because XP is efficient, or simply because the instructor teaching the course is very enthusiastic?
  (internal validity?)

Lessons Learned: Academic vs. Industrial

- The logistic of making XP work in an academic setting is trickier (scheduling problems etc.)
- No steady development pace – lots of spikes (when assignments are due)
  - students work on the most urgent not most important issues
- Students serve multiple "masters" at a time
- Fixed Scope – Flexible Time (assignments) and Fixed Scope-Fixed Time (exams) vs.
  Flexible-Scope - Fixed Time (in industry)
  - academic projects break the timebox

Lessons Learned: Communication

- Direct students to improve their English and communication skills
- Encourage electronic communication in the off-class time
- Discourage "assumptions disease" and the NIH syndrome
- Ask students to submit their estimates

What Did Not Work

- Scheduling pair programming sessions;
- Difficulties in adjustments when there was a big difference in skill levels;
- "Estimations were very poor for the first round"
- "Sometimes we had no idea where or how to start to solve the problem so building a test first design was difficult and frustrating".

http://www.agilenetwork.ca/study/

Employee satisfaction in agile teams

- World-wide survey of agile developers and managers
- **251** responses
- Compared to your other experiences of working in a non-agile team, how would you rate your current job now?
  - **47%** much better, **25%** better
- Level of job satisfaction
  - **29%** very satisfied, **55%** satisfied
- The two most important job satisfiers are:
  - Opportunity to work on interesting projects (**62%**)
  - Ability to influence decisions that affect you (**50%**)

Preliminary Results. Do Not Quote

Issues with agile methods

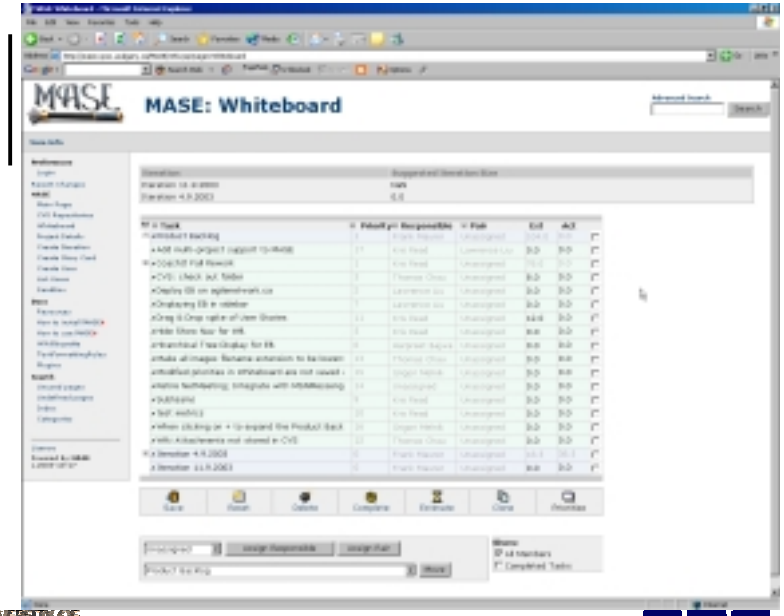
- Consultants have a short term perspective
- Anecdotal evidence often from people earning their money with agile methods
- Empirical evidence is weak

Empirical software engineering

- Empirical data is needed to support claims made by researchers
- Internal and external validity of published data questionable
- External validity needs industrial input
  - Tricky to control environment
  - How to convince companies without the data to participate in study
  - Late adopters play catch-up

Extending agile methods

- Tool support & distributed teams
  → MASE
- Scalability
  → COACH-IT
- Emerging design
  → Design Pattern Developer



Agile practices influenced by team distribution

- Planning game
  - Planning, coordination, tracking
- Information access and routing
- Built process
- On-site customer
- Knowledge sharing

Experiences

- System is used by development team
- Distributed progress tracking works
- User story management handy
- Iteration planning uses system even for collocated meetings

Distributed Pair Programming

"[Distributed] pair programming seemed to work really well. I would say that it was a little slower than pair programming normally is, however there was still a good flow of information (for example with the debugging tools, which I am by no means expert in). Additionally code quality seemed to be almost on-par with the pair programming I have done.

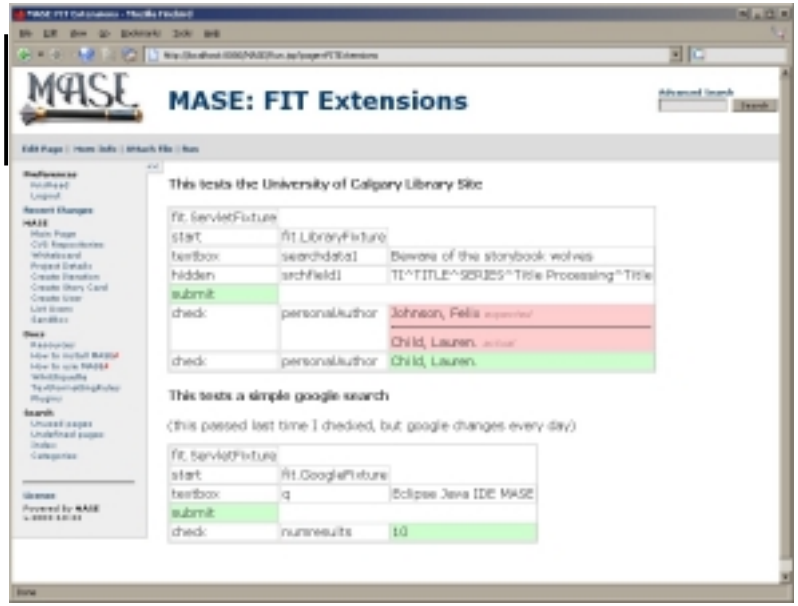
One of the factors slowing down the programming was that the audio was a bit halting at times, and there seems to be a tiny lag between when something was said and when it was heard. This doubtless cannot be helped due to latency."

K. Read

- Not big difference to collocated pair programming
- Speed fine using cable model and UofC connection
- Pointing to something more effort
- Problems:
  - Do other things on the side
  - JUnit UI not properly displayed via NetMeeting

MASE with COACH-IT

- Goal: Scaling agile practices
- Approach:
  - Divide and conquer strategy based on software architecture
  - Enforce contracts by test code: teams that use a component write tests for it
  - Use of web front end to define FIT tests



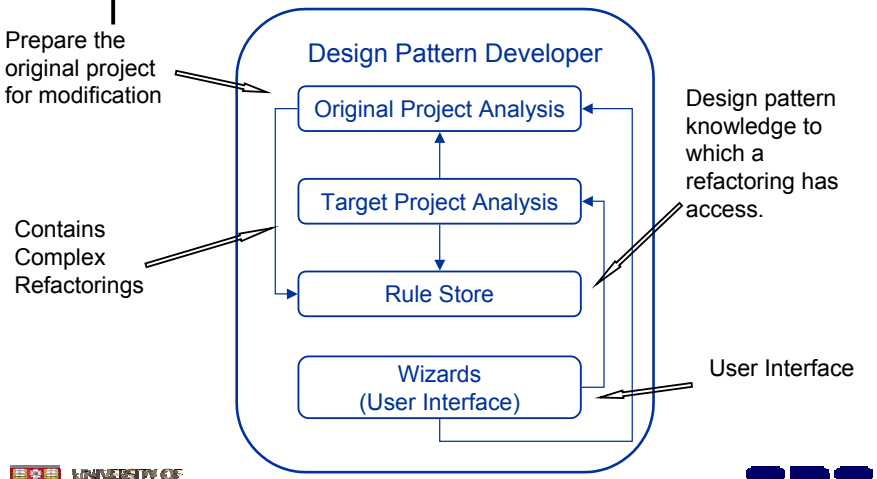
Emerging design

- Support Emerging Design to Design Patterns via:
  - Tool Support for Complex Refactoring
  - Results of Experimentation with the Tool

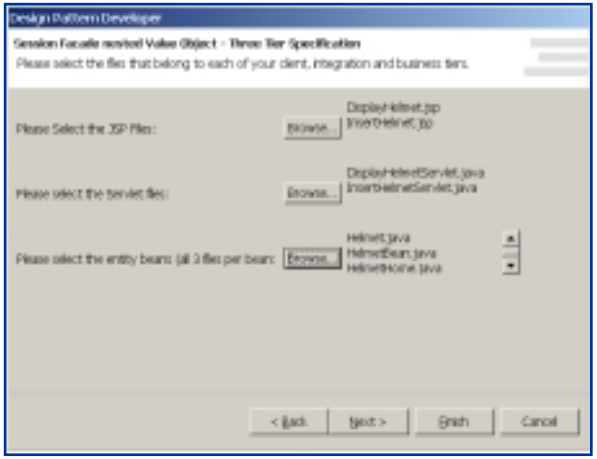
Refactoring Classifications

Refactoring	**Atomic**	**Sequential**	**Complex**
Original Source Scope	<= 1 class	1 field..* classes	>= 1class
Target Source Scope	< 1 class	0 .. * classes	0.. * classes
Nested Refactorings	Atomic	Atomic, Sequential	Atomic, Sequential, Complex
Knowledge	1 AST	*AST	*AST Design Patterns

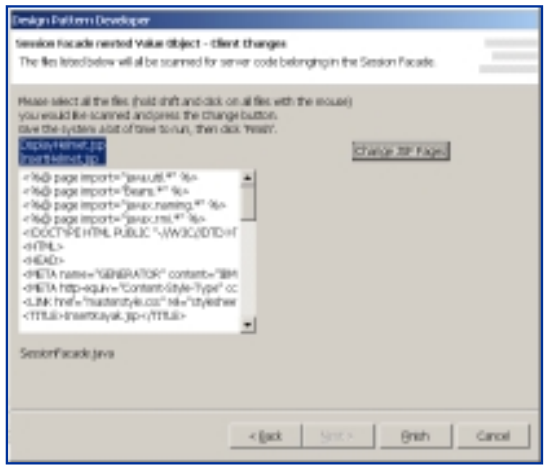
Design pattern developer



3. Example Tool Page

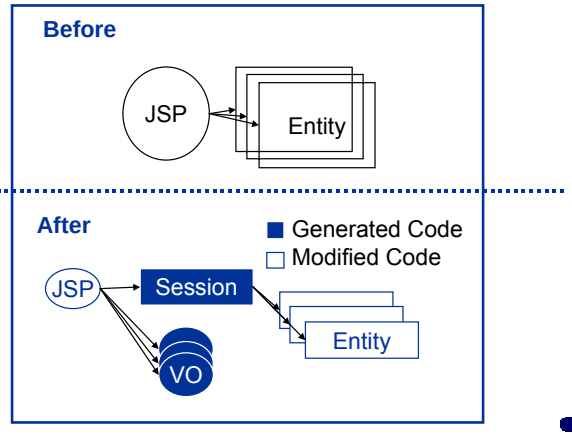


3. Example Tool Page

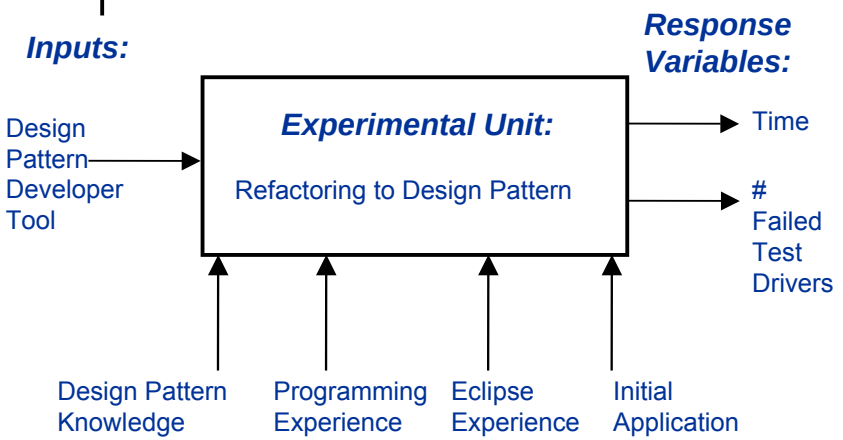


3. Example Tool Change

Session Façade nested Value Object Application



Explorative Case Study



4. Explorative Case Study

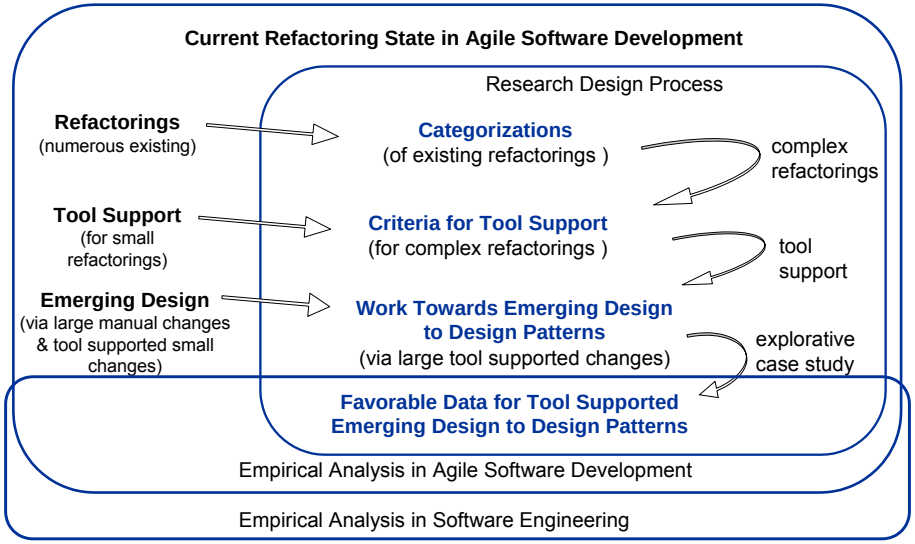
	Variables						
Participant	Tool Time	Completed (Test Drivers Passed)	Manual Time	Completed (Test Drivers Passed)	J2EE Experience	Design Pattern Experience	Tool over Manual
1	7	Yes	47	No	1 part time month	1 part time year	Yes
2	Did Not Complete Experiment						
3	24	Yes	48	No	3 years	4+ years	Yes
4	17	Yes	60	No	4 years	N/A	No
5	14	Yes	33	Yes	2 years	0	Maybe
6	7	Yes	N/A	No	< 1 year	1 year	Yes
7	17	Yes	35	No	1.5 years	6 months -1 year	Maybe
8	34	No	27	Yes	3 years	3 years	No
9	19	Yes	N/A	No	2.5 years	1 year	Yes

Explorative Case Study

- High & Low Lights:
  - "Manual. I come from the 'code crafter' school rather than the 'test and refactor' school. So, chances are I wouldn't need it on good code, but on bad code I'd worry about safety"
  - "[It] would depend on [the] situation. Probably use the tool and then manually adjust afterwards."
- Statistical Analysis of Time, using Paired T-Test:
  Null Hypothesis: $\mu_X = \mu_Y$
  Alternative Hypothesis: $\mu_X < \mu_Y$
  where μ is the mean time to refactor, X is the group that refactored with the tool and Y is the group that refactored without the tool.

  The null hypothesis is proved false with $\alpha = 0.5$

Research Process Overview



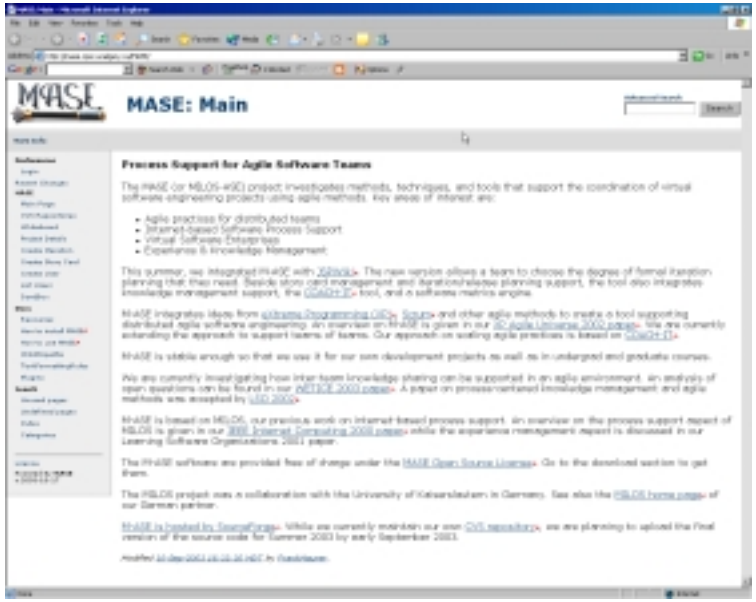
EBE Team

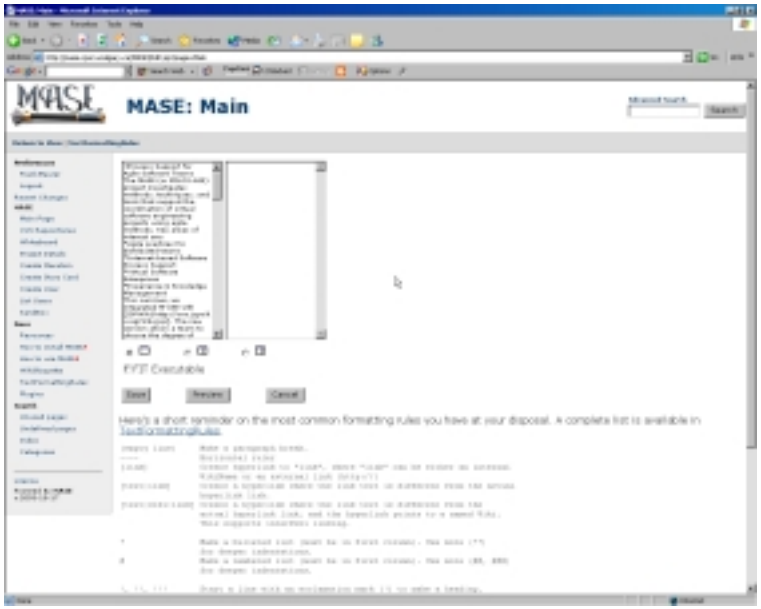
Credits:

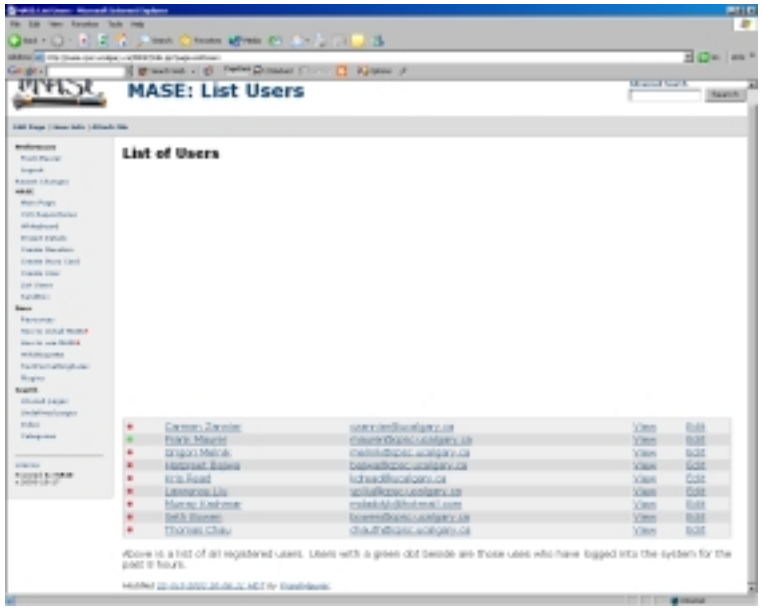
Empirical results	Grigori Melnik, Sebastian Martel
MASE Core	Harpreet Bajwa, Thomas Chau, Lawrence Liu, Kris Read
COACH-IT	Kris Read
MASE KM	Thomas Chau, Harpreet Bajwa, Lawrence Liu
Emerging design	Carmen Zannier
OBR	Philip Nour
P2P	Seth Bowen
TDD	Harpreet Bajwa, Chris Mann, Wenliang Xiong, Carmen Zannier
Lightweight KM	Lawrence Liu

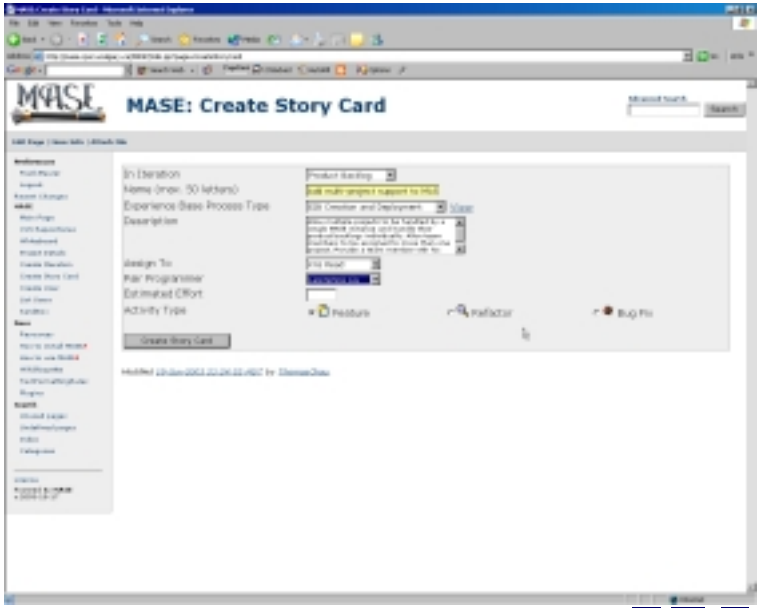
Research Web site: http://sern.ucalgary.ca/~milos

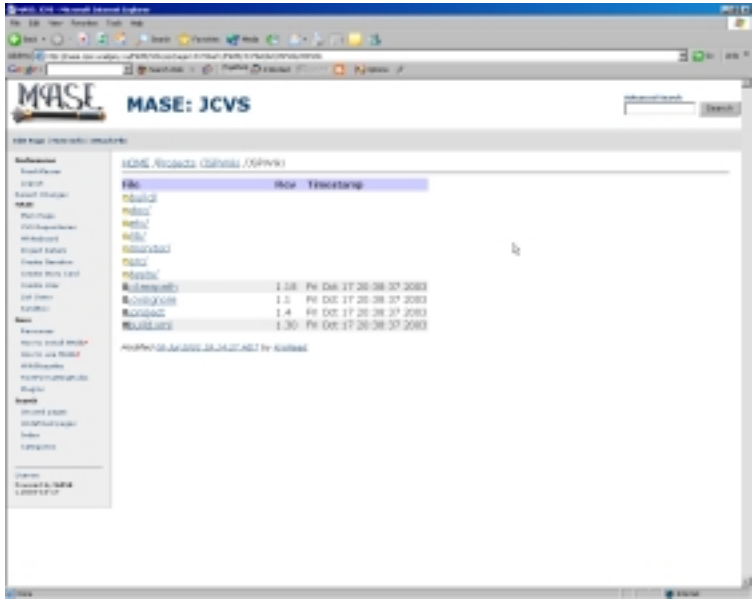
Collaboration: maurer@cpsc.ucalgary.ca

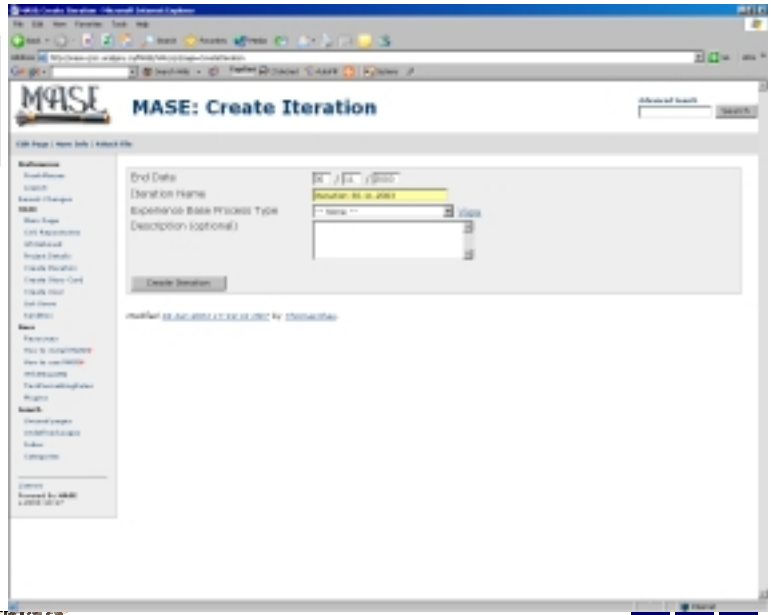


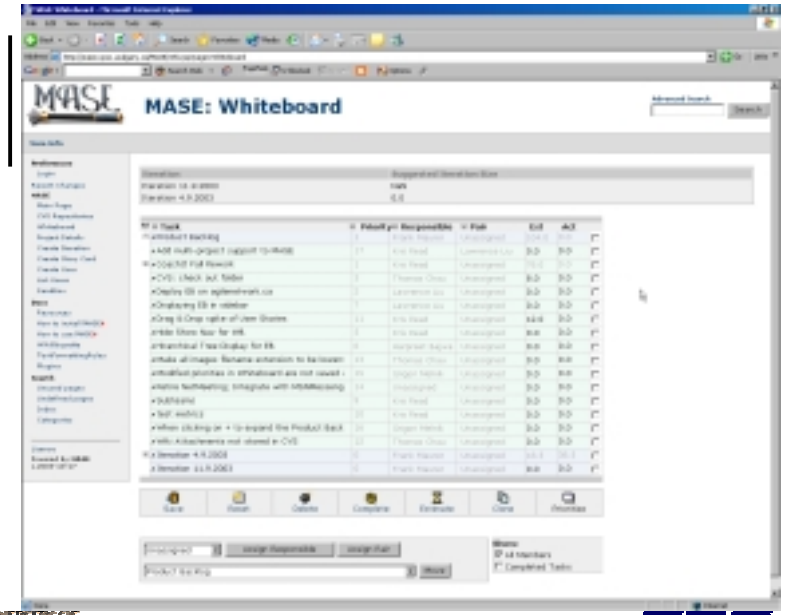


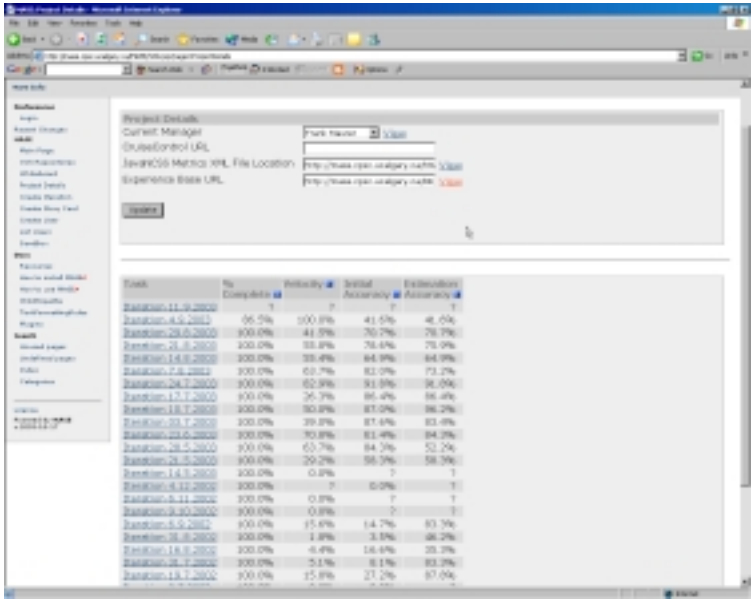


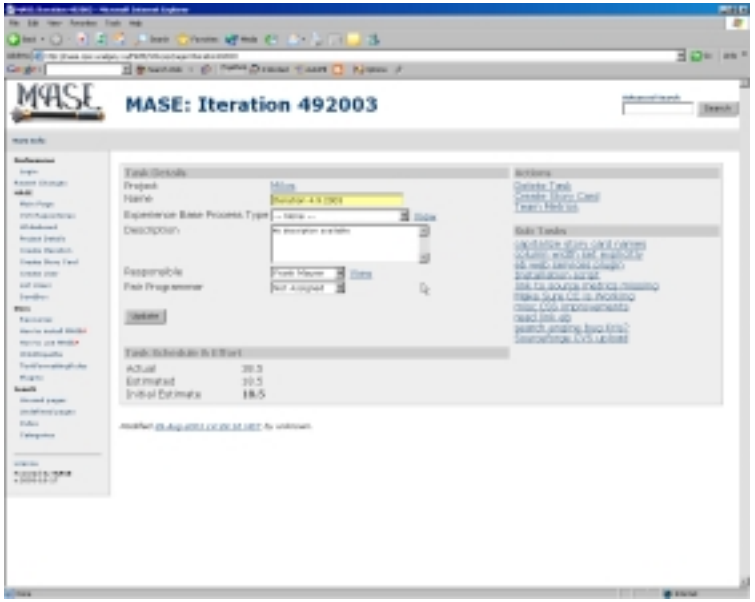

MASE: Create Story Card

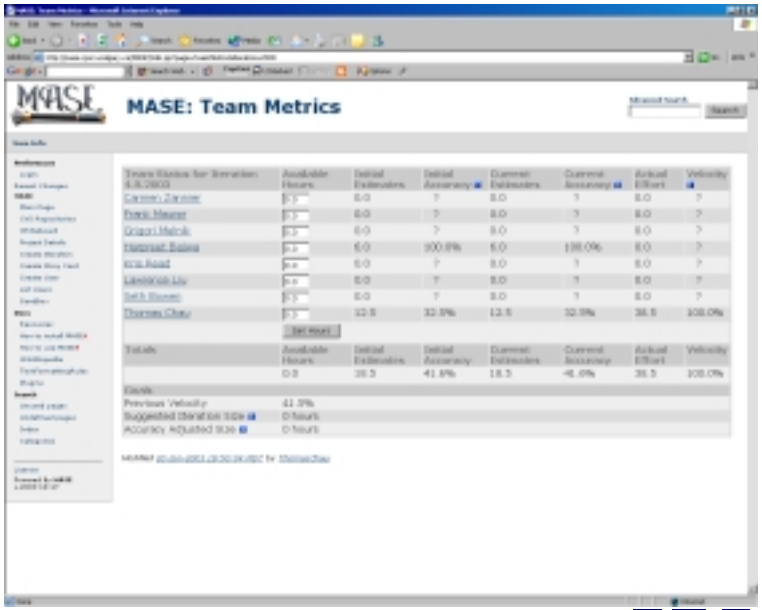

MASE: JCVS

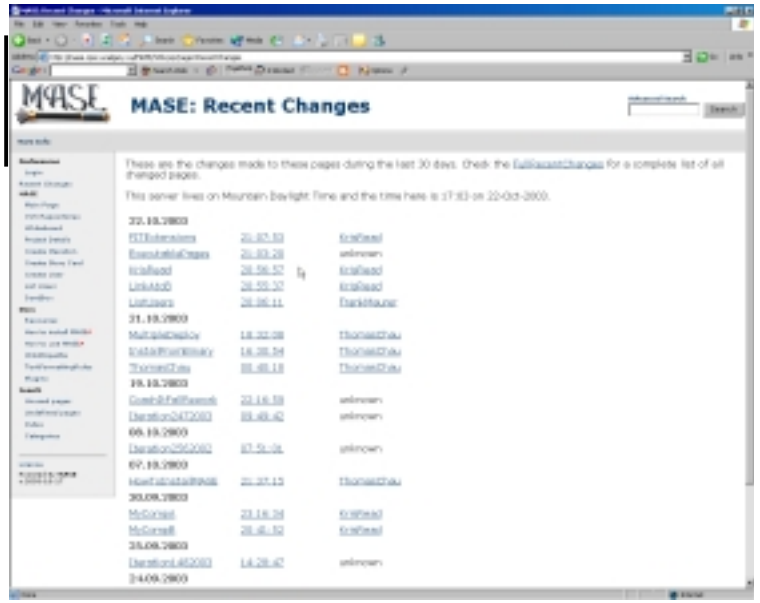

MASE: Create Iteration

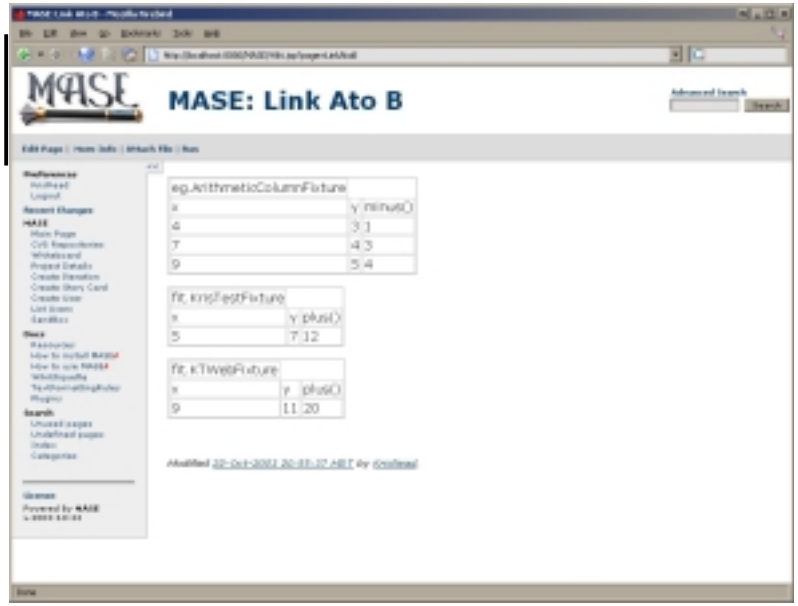

MASE: Whiteboard

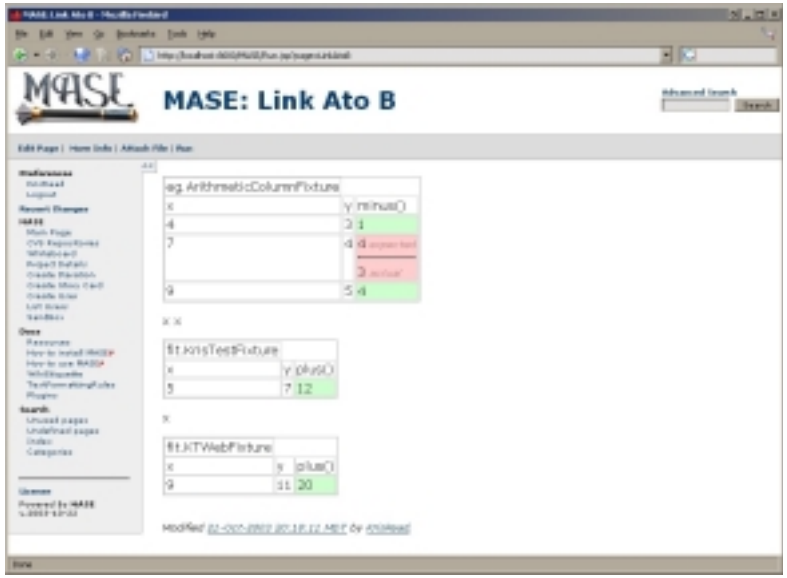



MASE: Iteration 492003

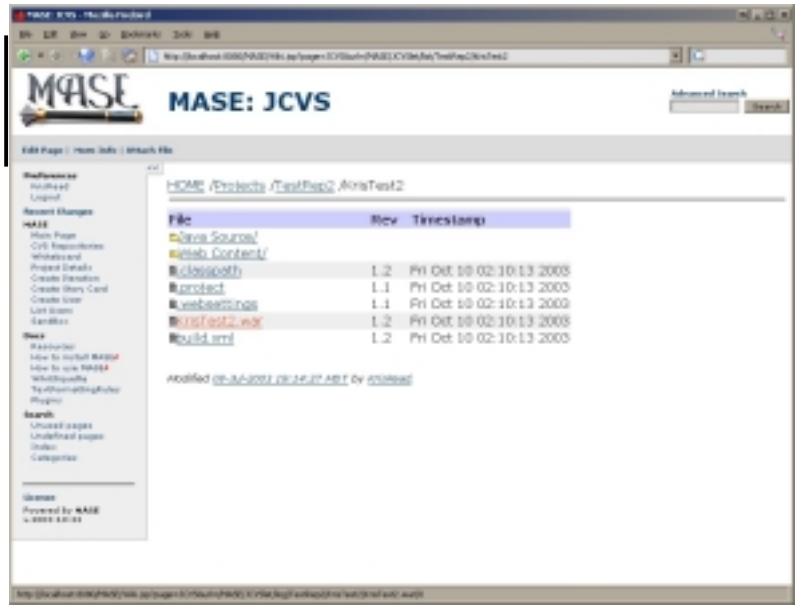

MASE: Team Metrics

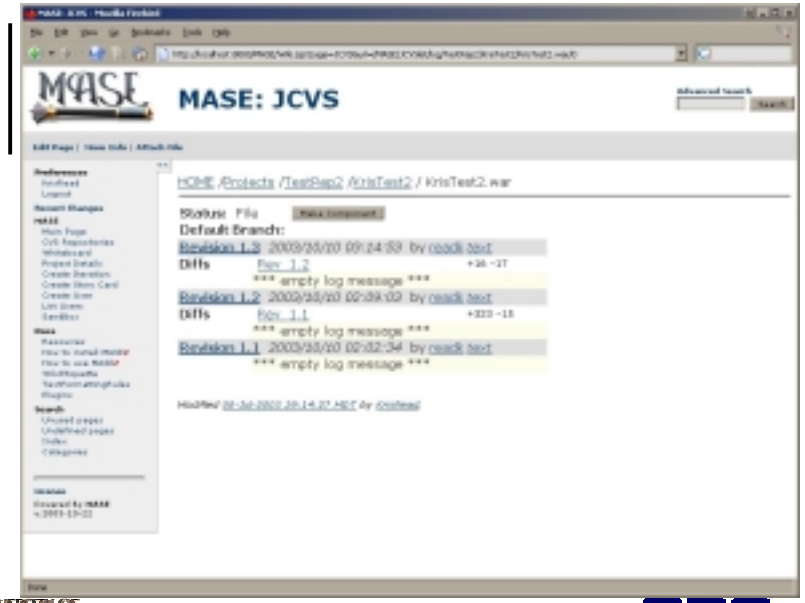

MASE: Recent Changes


MASE: Link Ato B
eg.ArithmeticColumnFixture
fit.KTWebFixture

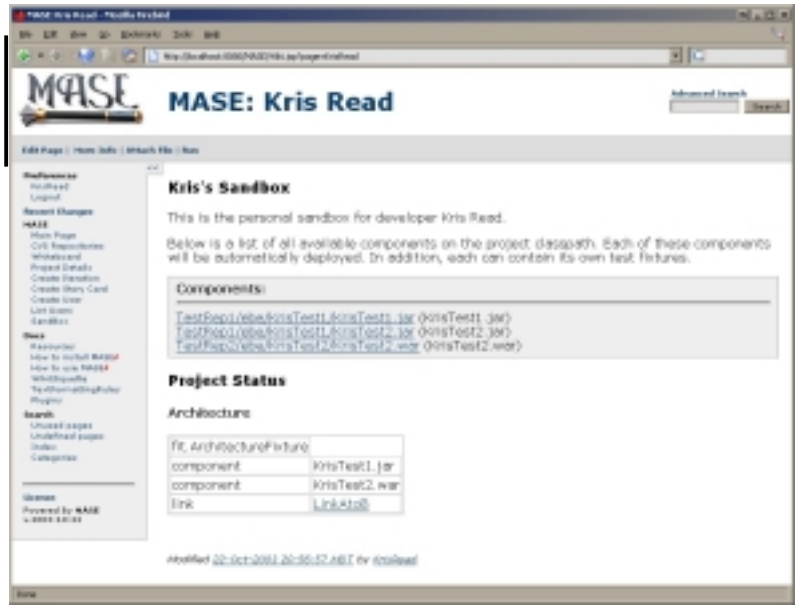

MASE: Link Ato B
eg.ArithmeticColumnFixture
fit.KTWebFixture

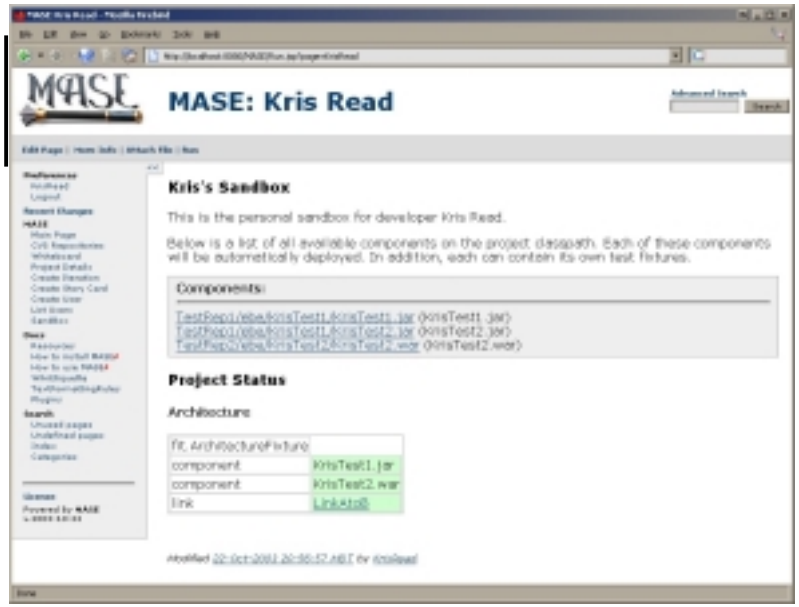
MASE: Link Ato B
FIT Executable
Here's a short reminder on the most common formatting rules you have at your disposal. A complete list is available in TextFormattingRules.

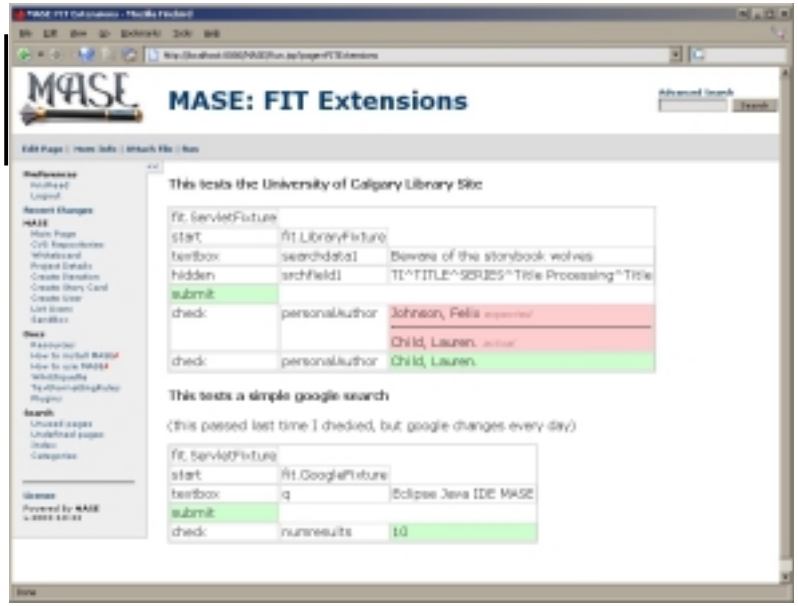

MASE: JCVS
HOME /Projects /TestProj2 /KrisTest2
File
Rev
Timestamp
Java Source/
Web Content/
.classpath
.project
.websettings
krisTest2.war
build.xml

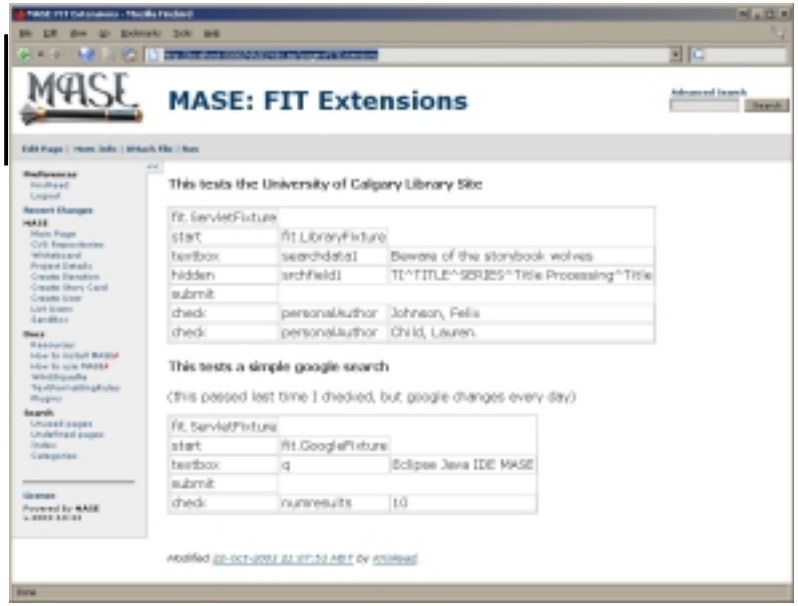

MASE: JCVS
HOME /Projects /TestRep2 /KrisTest2 / KrisTest2.war
Status: File
Default Branch:
Revision 1.3 2003/03/10 09:14:59 by readk Next
Diffs Rev 1.2
*** empty log message ***
Revision 1.2 2003/03/10 02:09:03 by readk Next
Diffs Rev 1.1
*** empty log message ***
Revision 1.1 2003/03/10 02:02:34 by readk Next
*** empty log message ***

MASE: JCVS
HOME /Projects /TestRep2 /KrisTest2 / KrisTest2.war
Status: Component KrisTest2.war
Default Branch:
Revision 1.3 2003/03/10 09:14:59 by readk Next
Diffs Rev 1.2
*** empty log message ***
Revision 1.2 2003/03/10 02:09:03 by readk Next
Diffs Rev 1.1
*** empty log message ***
Revision 1.1 2003/03/10 02:02:34 by readk Next
*** empty log message ***


MASE: Kris Read
Kris's Sandbox
This is the personal sandbox for developer Kris Read.
Below is a list of all available components on the project classpath. Each of these components will be automatically deployed. In addition, each can contain its own test fixtures.
Components:
TestRep1/ebe/KrisTest1.KrisTest1.jar (KrisTest1.jar)
TestRep1/ebe/KrisTest1.KrisTest2.jar (KrisTest2.jar)
TestRep2/ebe/KrisTest2.KrisTest2.war (KrisTest2.war)
Project Status
Architecture
fit.ArchitectureFixture
component KrisTest1.jar
component KrisTest2.war
link LinkAtoB


MASE: Kris Read
Kris's Sandbox
This is the personal sandbox for developer Kris Read.
Below is a list of all available components on the project classpath. Each of these components will be automatically deployed. In addition, each can contain its own test fixtures.
Components:
TestRep1/ebe/KrisTest1.KrisTest1.jar (KrisTest1.jar)
TestRep1/ebe/KrisTest1.KrisTest2.jar (KrisTest2.jar)
TestRep2/ebe/KrisTest2.KrisTest2.war (KrisTest2.war)
Project Status
Architecture
fit.ArchitectureFixture
component KrisTest1.jar
component KrisTest2.war
link LinkAtoB


MASE: FIT Extensions
This tests the University of Calgary Library Site
fit.ServletFixture
start fit.LibraryFixture
textbox searchdata1 Beware of the storybook wolves
hidden srchfield1 TI^TITLE^SERIES^Title Processing^Title
submit
check personalAuthor Johnson, Felix
Child, Lauren
check personalAuthor Child, Lauren
This tests a simple google search
(this passed last time I checked, but google changes every day)
fit.ServletFixture
start fit.GoogleFixture
textbox q Eclipse Java IDE MASE
submit
check numresults 10


MASE: FIT Extensions
This tests the University of Calgary Library Site
fit.ServletFixture
start fit.LibraryFixture
textbox searchdata1 Beware of the storybook wolves
hidden srchfield1 TI^TITLE^SERIES^Title Processing^Title
submit
check personalAuthor Johnson, Felix
check personalAuthor Child, Lauren
This tests a simple google search
(this passed last time I checked, but google changes every day)
fit.ServletFixture
start fit.GoogleFixture
textbox q Eclipse Java IDE MASE
submit
check numresults 10