What is software development at your company really like (methodologies, tools, ...)? [closed]

Asked 16 years, 2 months ago Modified 14 years, 2 months ago

Viewed 3k times

22

votes

Since I've started my first job as a professional software developer about two years ago, I've read many articles about commonly accepted methodologies (e.g. Scrum, XP), technologies (e.g. EJB, Spring), techniques (e.g. TDD, code reviews), tools (bug tracking, wikis) and so on in software companies.

For many of these I've found that we at our company doesn't use them and I ask myself why. Are we doing it wrong or is it merely that these articles I've read are not really telling what it's like in the real world? Are these articles more academic?

So, please tell me what it's like at your company. Tell about everything regarding software development. Here are some suggestions (in the order as they come from my mind). Tell at least if you do it or not, or give a short comment:

- Test-Driven-Development
- Domain-Driven-Design
- Model-Driven-Design/Architecture
- Do you test?
- Unit Testing
- Integration Testing
- Acceptance Testing
- Code Reviews
- Innovative Technologies (Spring, Hibernate, Wicket, JSF, WS, REST, ...)
- Agile
- Pair Programming
- UML
- Domain-specific languages
- Requirement Specification (How?)

- Continous Integration
- Code-Coverage Tools
- Aenemic Domain Model
- Communication (Wiki, Mail, IM, Mailinglists, other documents)
- Effort estimates
- Team size
- Meetings
- Code metrics
- Static code analysis
- Bug tracking
- ...

And remember: I want to know what you really do, not what you would like to do or think you should do.

survey

Share

edited Oct 23, 2008 at 12:34

Maxime Rouiller

13.7k ● 9 ● 60 ● 108

asked Oct 23, 2008 at 11:59

cretzel

20.1k ● 19 ● 60 ● 73

2 I actually Don't know... I do whatever my boss tells me to do :) – THEn Jun 18, 2009 at 19:57

1 This should be CW, there is no single answer to this question – Pascal Thivent Nov 1, 2009 at 19:26

Comments disabled on deleted / locked posts / reviews

23 Answers

Sorted by: Highest score (default)

26 votes

- **Test-Driven-Development** - No way.
- **Domain-Driven-Design** - What's *design*?
- **Model-Driven-Design/Architecture** - What's *design*? We do have an architecture team. With one exception (the most junior architect), they couldn't code their way out of a paper bag. They're sure good at drawing boxes with lines, though! And establishing crappy, worthless, over-generic and completely useless standards. (The old OPC stuff is OK, but the UA standard has been "done next month" for the last 4 years or so.)
- **Do you test?** - Yep, we do have a dedicated test team. There's about 1 tester for every 10-12 devs. They're completely swamped. Ask me if we test *well*.
- **Unit Testing** - Completely informal/up to the developer. I do when the schedule I'm given allows for it.
- **Integration Testing** - Yes. This one's necessary given the suite of products we develop and support.

- **Acceptance Testing** - Yes, for contract-y work only.
- **Code Reviews** - Always pay lip service, never *ever* do it.
- **Innovative Technologies (Spring, Hibernate, Wicket, JSF, WS, REST, ...)** - Taking new dependencies is strongly frowned upon. Boost will never be adopted, e.g. We have generally had good luck getting to newer versions of .Net, though, if typically 2 years or so behind the curve.
- **Agile** - No. Management claims to want "agile," though they don't exhibit the barest understanding of what it is. We just recently modified our process so that higher priority tasks are spec'd and implemented with... (wait for it) higher priority! Management tells me that this is our new "agile" process. It still smells, walks, and quacks like a waterfall though.
- **Pair Programming** - No way! Pay *two* people to do the work of one? Next you'll be suggesting that developers should waste time on nonsense like *designs* and *code reviews*. Dogs, cats, living together!
- **UML** - No. We got a UML tool once to help us understand a legacy codebase that had evolved. The person in charge of evaluating the tool loved it, it reverse engineered the entire million+ line C++ codebase in less than 30 seconds! After they were talked into buying it and actual devs tried to use it, we found that it really just took those 30 seconds to fail to parse 95+% of the codebase. The error reporting was so bad the evaluator hadn't even figured out that it

failed. (I'm lookin' at you, kid!) It only took us a year and a half to get around to dropping our licenses for that. See? Agile!

- **Domain-specific languages** - They're probably used somewhere, though not by myself.
- **Requirement Specification (How?)** - A product manager performs some voodoo and invents them. Sometimes they may even talk with customers about them! If you're really lucky, they'll even understand the difference between a use case and a requirement. Don't count on it, though. We don't really do use cases.
- **Continous Integration** - No way. It's far more exciting when everything breaks at once.
- **Code-Coverage Tools** - Someone once put a blankey on the source repository server in the cold, cold server room. Does that count?
- **Aenemic Domain Model** - In all seriousness, I've never even heard of this before.
- **Communication (Wiki, Mail, IM, Mailinglists, other documents)** - Memos. Lotus Notes doesn't *do* "e-mail". Bunch of newfangled rubbish.
- **Effort estimates** - Not really. In my organization, Estimates are code for targets.. The due date for a project is locked in during the first of the project's 5 "agile" phases of waterfall development. Those due dates are called "ballpark estimates" but really mean "ship dates."

- **Team size** - Runs the gamut, based on product. We have teams as small as four and as big as fifteen if you include managers.
- **Meetings** - Not bad if you're relatively junior and aren't working on more than one or two products. I'm only required to attend 2-3 1-hour meetings per week.
- **Code metrics** - No.
- **Static code analysis** - Theoretically for .Net b/c FxCop is built in and it's use is mandated by our standard, but really, no. Nobody checks it b/c there are never any code reviews. Just the occasional quality audit (aka, paper-trail/blame audit) to make sure we don't lose whatever this year's certification is.
- **Bug tracking** - Yes, but only for customer-reported problems. Developers are not allowed to submit discovered bugs against a product they're working on b/c that's not being a "team player." (My boss' boss explained this to me in great detail when I made *that* mistake. I'm now friendly with a particular customer who's willing to "discover" bugs that I might "accidentally" mention in the course of other support-related communication.)

As far as big, corporate dev't goes, there's a lot worse out there. Given where I live, and the lack of high-tech jobs in the area, I'm actually pretty lucky to have a gig at all. Doesn't mean I have to like the way things are, though. It just takes a lot of time and constant pressure to even try to influence an established corporate culture.

But if they get sick of my attempts to change the culture and fire me, well, I don't think I'd cry myself to sleep that night.

Share

answered Oct 23, 2008 at 13:18

Greg D

44.1k ● 14 ● 87 ● 118

1 "I don't think I'd cry myself to sleep that night." - sounds like that might depend how many tequila shots you did at your celebration party. – Steve Jessop Oct 23, 2008 at 13:52

10 Hi do we work together? :) – Nick Stinemates Jan 10, 2009 at 20:22

2 My favourite: "See? Agile!" xD – Arnis Lapsa Jun 18, 2009 at 13:18

1 But then i read: "Hi do we work together? :)". – Arnis Lapsa Jun 18, 2009 at 13:19

1 Who wants to come to my celebration party? :D I'm doing shots of tequila. I just learned that the org is getting out of software dev't at this site, presumably because (shockingly!) we can't turn a profit on software that's written here. – Greg D Jun 18, 2009 at 13:27

5

votes

I think the famous *Big Ball of Mud* pattern describes a lot of work environments and gives you some good ideas about how to combat this kind of thing.

By the way, I realize I'm not directly answering your question but Big Ball of Mud prevails in a depressingly

large percentage of development situations. You can ask about test driven development and defect tracking and other sorts of things of that sort but if the truth is told from what I've seen, I'd say the Big Ball of Mud is pretty much the de facto way that people work--whether they should or should not.

Share

answered Oct 23, 2008 at 12:45

Onorio Catenacci

15.3k ● 16 ● 84 ● 134

2 Big Ball of Mud - (maybe it should be renamed "entropic heat death"?) The sad thing is when people have done nothing but contribute to the mudball their entire working lives, and don't think there's anything wrong with it. – Steve B. Jun 18, 2009 at 13:41

Most of my co-workers believe that it's not possible at all to code properly. I feel much more idealistic. :) – Arnis Lapsa Jun 18, 2009 at 13:50

1 yeah, same here, Big Ball of Mud, except of course MY code ;) – Andriy Volkov Nov 1, 2009 at 19:30

2

votes

- Test-Driven-Development - Almost there.
- Domain-Driven-Design - No
- Model-Driven-Design/Architecture - No
- Do you test? - Yes
- Unit Testing - Yes
- Integration Testing - Yes

- Acceptance Testing - No
- Code Reviews - No
- Innovative/New Technologies (Spring, Hibernate, Wicket, JSF, WS, REST, ...) - ASP.NET MVC? NHibernate? Yes
- Agile - Just started
- Pair Programming - No
- UML - Nothing formal
- Domain-specific languages - No
- Requirement Specification (How?) - Yes. Capturing story requirements.
- Continous Integration - Yes. TeamCity
- Code-Coverage Tools - Yes. NCover
- Aenemic Domain Model - No
- Communication (Wiki, Mail, IM, Mailinglists, other documents) - IM, Email
- Effort estimates - 1,2,4,8
- Team size - 4
- Meetings - Daily stand up
- Code metrics - No
- Static code analysis - No
- Bug tracking - Existing custom job

My department is a work in progress. Over the past few months, I've made an effort in enforcing continuous

improvement. Some stuff has been down right difficult to talk about. However, when looking back, they have improved.

Share

answered Oct 23, 2008 at 12:35

Shane Bauer
263 ● 1 ● 4 ● 13

2

votes

- Test-Driven-Development: yes
- Domain-Driven-Design: yes
- Model-Driven-Design/Architecture: yes
- Do you test? yes
- Unit Testing - yes
- Integration Testing - yes
- Acceptance Testing - yes
- Code Reviews - yes
- Innovative Technologies - yes
- Agile - solely agile
- Pair Programming - yes
- UML - sometimes for ddd whiteboard fights
- Domain-specific languages - yes
- Requirement Specification (How?) - in the form of examples and acceptance tests
- Continous Integration - yes - TeamCity
- Code-Coverage Tools - yes - NCover

- Aenemic Domain Model - not sure
- Communication (Wiki, Mail, IM, Mailinglists, other documents) - wiki, mail, msn
- Team size - 6+ dependent on project
- Meetings - every morning at 9:30 - SCRUM
- Code metrics - dunno
- Static code analysis - dunno
- Bug tracking - Mantis

And most importantly...

- Everyone goes home at 5:30: **YES**

However the salary is alot lower because alot of people want to work for this company. Can't have everything!

Share

answered Dec 8, 2008 at 13:52

Nick
51 ● 3

2 Interesting observation. So if a company provides very nice place to work, they can save money on wages :-) – Kugel Aug 17, 2010 at 17:22

@Kugel: Absolutely true. I've observed this more than once. – Greg D Oct 4, 2011 at 13:52

2 votes

- **Test-Driven-Development:** Nope. At best, in very tiny portions. We're all talking about, but don't do it.


- **Domain-Driven-Design:** Nope. Hard to know what a "domain" is if you're developing a technical framework. Have not much experience in DDD to know how to do it.
- **Model-Driven-Design/Architecture:** Nope.
- **Do you test?:** Yes, but not enough. With every release (we're trying to push out minor releases every 8 weeks) there're always more than 2 service releases. We're in the first year of product development, so i think this is pretty okay.
- **Unit Testing:** Yes. At approx. 30% coverage. Most of the developers know now that they should write unit tests for themselves. Every time they have to fix a critical bug in their own code, they can see the benefit if they would have written a simple test up front to prevent the bug in the first place.
- **Integration Testing:** Yes, using Selenium and WebDriver.
- **Performance Testing:** Yes, beginning with now. Goal is to archive long-term performance measurements and compare them against releases. Using JMeter and a dedicated performance test server for that.
- **Acceptance Testing:** Not really, but our product is used internally too and we're getting feedback pretty fast before it's being released. I count that as acceptance testing.
- **Code Reviews:** Nope. Sometimes, someone else looks at it for 30 minutes, but that's it.

- **Innovative Technologies (Spring, Hibernate, Wicket, JSF, WS, REST, ...):** From my POV, those technologies are not "innovative" any more. They're pretty much old-school now. Except JSF, which died a couple of years ago. Did Spring+Hibernate for the last couple of years. Now doing Wicket + WS for 2 years. Replaced Hibernate with iBatis SqlMaps.
- **Agile:** Nope.
- **Pair Programming:** Nope.
- **UML:** A little bit, mainly for deployment diagrams. Class diagrams too fine-granular and often are not in sync with reality. Developers do what they think is best, not what an UML diagram tells them to do.
- **Domain-specific languages:** Yes, we're using home-brewn business rules technology. It's a visual DSL which is suitable for endusers. Kind of like using Visio to model decisions.
- **Requirement Specification (How?):** Nope.
- **Continuous Integration:** Yes. Based on Hudson and Maven. Unit tests are run on each build. Additional nightly builds with more detailed reporting enabled. Whole team is notified about failed builds (yeah, they complain about getting too many mails sometimes if something breaks the chain and all 30 submodules get build failures, e.g. when the Maven Repository is unreachable)
- **Code-Coverage Tools:** Yes. Maven/Cobertura and Sonar.

- **Aenemic Domain Model:** No idea what this is supposed to be.
- **Communication (Wiki, Mail, IM, Mailinglists, other documents):** Wiki, Mail, IM, Daily standup meetings, Enduser and developer manuals written by a professional/non-developer.
- **Effort estimates:** Trying hard to do good estimates. But without reqs, they are just rough estimations. Good enough for resource planing though.
- **Team size:** 12
- **Meetings:** Daily standup, retro every 8 weeks after each minor release.
- **Code metrics:** Sonar. Looking to comply with most of the rules. Did not have time to reconfigure the rules to suit our needs though.
- **Static code analysis:** Sonar.
- **Bug tracking:** JIRA

Notes:

- Sonar is a code quality server. It combines tools like PMD, Findbugs, Checkstyle etc.

Share

answered Nov 1, 2009 at 22:15

mhaller

14.2k ● 2 ● 43 ● 62

1

- Test-Driven-Development - No

vote

- Domain-Driven-Design - No
- Model-Driven-Design/Architecture - No
- Do you test? - Almost never
- Unit Testing - Almost never
- Integration Testing - No
- Acceptance Testing - No
- Code Reviews - No
- Innovative Technologies (Spring, Hibernate, Wicket, JSF, WS, REST, ...) - Spring, Hibernate, Wicket
- Agile - No
- Pair Programming - No
- UML - just sketches
- Domain-specific languages - No
- Requirement Specification (How?) - We get a huge customer requirement specification and we use mind maps to extract the actual features which are then estimated
- Continous Integration - No
- Code-Coverage Tools - No
- Aenemic Domain Model - Yes
- Communication (Wiki, Mail, IM, Mailinglists, other documents) - Mind maps, Mail
- Effort estimates - FITA (Finger in the air, see here)
- Team size - 2-6

- Meetings - 2-3 times a week
- Code metrics - No
- Static code analysis - No (Tried FindBugs and Checkstyle)
- Bug tracking - Yes, Bugzilla

Share

answered Oct 23, 2008 at 12:00

cretzel

20.1k ● 19 ● 60 ● 73

2 Hey you must work where I work...Or else methodologies don't hold up under tight deadlines and money pressure. – nportelli Oct 23, 2008 at 12:29

Note there's a difference between a methodology "not holding up", i.e. you try it and abandon it, and never trying it in the first place. For example, most places manage to do at least some testing, so if you do none I'm not sure you can blame that on testing being a worthless methodology... – Steve Jessop Oct 23, 2008 at 12:40

Allmost the same here ... And guess what our software calculates billions and actually with no bugs (at least found by end-user testers)... Anyway just watch for the following phrase "No way this shit ain't ever goig to work " from youtube.com/watch?v=Pug_5Tl2UxQ – Yordan Georgiev May 1, 2009 at 17:36

1 vote

I feel sorry for you :) It's not a good environment to work in, as you need to constantly exercise practise good practices to really understand and use them.

I know several (mine included) companies which would be able to tick all the '*good*' boxes in your list.

However the devil is in details and even in some companies with good SDP policies not every project follows them.

Share

edited Oct 23, 2008 at 15:12

answered Oct 23, 2008 at 12:19

Ilya Kochetov

18.4k ● 6 ● 47 ● 62

Are you sure you want to tick all the boxes? I didn't know what Anemic Domain Model is prior to just looking it up on Wikipedia, but allegedly it's a *bad* thing... – Steve Jessop Oct 23, 2008 at 12:37

I wouldn't want to tick all the boxes. I'll take established, tested, and well-supported technology over innovative technology any day. But then, my first development job was at a company which provided 5 9s uptime on our software, so what do I know? – tloach Oct 23, 2008 at 12:41

I think the only 5 9's system I've ever had the pleasure of using is a wristwatch :-) – Steve Jessop Oct 23, 2008 at 12:52

@onebyone: how about an ATM? The hardware may fail, they may be taken offline to fill up the cash bins or change the advertising displayed, but it's really, really rare to see a software error on one. It happens, but given the number of ATMs out there it's incredibly rare. – tloach Oct 23, 2008 at 12:54

My bank's whole ATM network went down for a morning a while back, due to a fault in the LINK system. That blew their 5 9s budget for the next (um...) 50 years or so. Sure, the coders

responsible for the components that didn't fail can say "not me, guv", but from a customer's POV the system failed.
– Steve Jessop Oct 23, 2008 at 13:06

1
vote

- Test-Driven-Development - if by this you mean writing tests before code, not always
- Domain-Driven-Design - not pure DDD
- Model-Driven-Design/Architecture - never, but really NEVER, again
- Do you test? - yes, always
- Unit Testing - yes, always
- Integration Testing - it depends but we try to avoid them as they are typically slow
- Acceptance Testing - yes, ideally automated
- Code Reviews - yes, this is included in the definition of done
- Innovative Technologies (Spring, Hibernate, Wicket, JSF, WS, REST, ...) - not sure the mentioned technologies are innovative but yes to Spring, Hibernate, WS
- Agile - yes, this is in my DNA
- Pair Programming - not always but yes (on new subjects, with new team members, if explicitly asked)
- UML - a little (i.e. a class or a sequence diagram on a whiteboard from time to time), only if helpful
- Domain-specific languages - no real usage until now

- Requirement Specification (How?) - lightweight specifications (e.g. user stories)
- Continous Integration - of course (and according to our definition of done, the code must have been "integrated")
- Code-Coverage Tools - yes (cobertura), this is in the definition of done
- Aenemic Domain Model - no, we try to avoid that
- Communication (Wiki, Mail, IM, Mailinglists, other documents) - face to face, wiki, IM, mail, mailing list (but we try to avoid word documents)
- Effort estimates - yes, in story points at the backlog level, in hours at the iteration level
- Team size - 7+/-2
- Meetings - yes, but only useful one and always time boxed (iteration planning, daily meeting, demo and retrospective)
- Code metrics - yes (cyclomatic complexity, code coverage, coding standards collected in sonar)
- Static code analysis - yes (findbugs, checkstyle)
- Bug tracking - yes, of course (but we try to fix bugs as soon as we discover them)

Share

answered Nov 1, 2009 at 19:51

Pascal Thivent
570k ● 140 ● 1.1k ● 1.1k

1

vote

- Test-Driven-Development - Though it should be as this was attempted to be brought in but I don't think it has taken off, so this is still a no but with more details now.
- Domain-Driven-Design - No
- Model-Driven-Design/Architecture - No
- Do you test? - Yes, but not comprehensively. We do have some unit tests, some integration tests and some WatiN tests.
- Unit Testing - We have some for our new development but the legacy ones don't.
- Integration Testing - Usually, when it is applicable. My team being the web team doesn't seem to have this too often yet.
- Acceptance Testing - We have a few levels of this. The first is when a new feature is being developed and has to get an initial approval from someone on another team that will be entering the content that comes before it is even integrated in with the code. The second is when the features get demonstrated at the end of a Sprint to get more feedback about what isn't looking right or working well. Then there is a third level just before it goes into production as a final, "Yes, this doesn't mess up what we have already," sort of thing.
- Code Reviews - These aren't done anymore but would probably be a good thing to do.
- Innovative Technologies (Spring, Hibernate, Wicket, JSF, WS, REST, ...) - There are some RESTful ideas

being applied in our project and we are using some features of the .Net framework like lambda expressions.

- Agile - We use Scrum and have stand ups, story board, Iteration Planning Meeting (That is really for the sprint and not an iteration which is 2 sprints as after each pair of sprints the work is shown to executives and other departments while the other demo is for an architect and the head of the content entering team.)
- Pair Programming - We do have pairing on most new development that isn't seen as grunt work. So for whoever wants to work on the Training part of the site, a pair will do it instead of just one developer.
- UML - No, and the tool for UML was removed in our new machines
- Domain-specific languages - No, but there is some terminology that is the company's own interpretations of things as some names of internal products bump against terms others may use for various things.
- Requirement Specification (How?) - This can range from a big word document spelling out what needs to be done to conversations of what to do and then try this and try that afterward.
- Continous Integration - We have Cruise Control.Net running for our CI that is used when we commit code changes.
- Code-Coverage Tools - Nope.
- Aenemic Domain Model - Somewhat in that there isn't really a big domain model here.

- Communication (Wiki, Mail, IM, Mailinglists, other documents) - In order of importance: E-mail, IM, phone, then visiting cubicle. There is a weekly team meeting with the manager of applications and daily standups on the big project.
- Effort estimates - This is now common in each sprint though sometimes this is done by sending out a spread sheet for everyone to put in their estimates that the Scrum Master combines all the results for us to see in the end.
- Team size - 5 developers with a team lead, a business analyst who is the Scrum Master, a tester to oversee what we have and others outside the team that pop up as needed including content authors to actually use the system.
- Meetings - Down to business, short, effective and typically good for communicating where things are currently.
- Code metrics - None that I know.
- Static code analysis - Nope.
- Bug tracking - Quality Center is used for tracking defects. *** Source Control - We are using Subversion now. For any feature or bug we create a new branch so we can work independently and not have our commits break the build as we are working on something. However, we all share the same DB for development which can be interesting at times.**

- IDE - Visual Studio 2008 on XP using .Net 3.5 and Sitecore 6.1
- ...

The team is on our 3rd team lead in the almost 2 years I've been here.

The CMS project is the big project that we are all working on though there are various support requests that come in that others handle.

There have been a lot of changes in the year that we've had a VP of IS. Production is more locked down and there is more work to get a release done as there is a check list procedure now and more hoops that may be useful.

Share

edited Nov 2, 2009 at 15:28

answered Oct 23, 2008 at 15:38

JB King

11.9k ● 4 ● 40 ● 49

1

vote

I am one of two programmers at a small software firm with VERY non-technical owners ("what's a 'browser'" - seriously asked last week).

The advantage is that I can choose for myself on most of these points.

Test-Driven-Development - Our owner had a bad experience or something. Mention testing in the wrong way and I'd swear she's acting out of PTSD.

Domain-Driven-Design - Yep.

Model-Driven-Design/Architecture - Yep.

Do you test? - Nope. Testing falls on the sales & support staff and the owners. So nothing gets tested much once it leaves my dev machine.

Unit Testing - If I got caught doing it I might get fired. And its seriously the only thing that could get me fired.

Integration Testing - See "Do you test?"

Acceptance Testing - Well, we have to deploy it sometime, right?

Code Reviews - No one else would understand it. I've seen everyone elses. Wish I hadn't.

Innovative Technologies (Spring, Hibernate, Wicket, JSF, WS, REST, ...) - Yep. But I get flak for it. I'm the "kid" who doesn't know anything that wasn't invented in the last ten years (despite being 30 and having "Readings in Database Systems" on my desk).

Agile - I don't waterfall. But I don't really Agile, either.

Pair Programming - You don't want to try to talk to the other "programmer" that works at our company.

UML - Nope. But I draw boxes with identifiers in them sometimes on my whiteboard. The bosses like that. Good thing they're not more technically inclined, or I'd probably see more boxes.

Domain-specific languages - Nope.

Requirement Specification (How?) - Nope.

Continous Integration - Nope.

Code-Coverage Tools - Nope.

Aenemic Domain Model - Yep. Tried it. Most of my situations I don't like it for and don't use it.

Communication (Wiki, Mail, IM, Mailinglists, other documents) - Tried, couldn't get coworker buy-in. Our MediaWiki install still has the default logo graphic.

Effort estimates - We have to estimate how long every job will take in hours. That is what the client gets billed. And that is what we are expected to spend on the project. We even do this when we are looking at new clients and developing new apps from scratch, as well as when we do bug fixes (yeah, we charge clients for that), feature additions, etc.

Team size - 1. And let me say this is not a good way to work. It is much better to be able to bounce ideas of other capable programmers in real time.

Meetings - few hours worth a week, sometimes double that and rarely less than that. Half the meetings I do are with

clients, have are totally internal.

Code metrics - Nope.

Static code analysis - Nope.

Bug tracking - Not as much as I should.

Share

answered Dec 5, 2009 at 4:10

quentin-starin
26.6k ● 7 ● 70 ● 87

1

vote

I work for a Ruby on Rails consultancy in Brisbane, Australia.

- **Test-Driven-Development**: This is pushed very, very hard in our office. Not testing is viewed as "incredibly stupid". You write the code, how do you ensure by way of an automated process such as CI, that it still works? Tests.
- **Do you test?**: See point one.
- **Unit Testing**: All the time, using RSpec. I'm "fluent" in Test::Unit and Shoulda also.
- **Integration Testing**: Again, all the time, Cucumber.
- **Acceptance Testing**: With our tickets we "deliver" them with an Acceptance Criteria. Then the client has to either "Accept" or "Reject" them by following the bouncing ball. The Acceptance Criteria has the bonus of also being what our Cucumber features are written in.

- **Code Reviews && Pair Progamming**: We pair. It's the instant version of code review. It's awesome because you can sit back and watch someone else work, they write the test and then you write the code to make that test pass. If you're sick then the other person knows what you were up to and can pair with somebody else.
- **Innovative Technologies**: Because we use Rails, we're really fond of REST.
- **Agile**: We work on 2 week iterations using Pivotal Tracker.
- **Requirement Specification (How?)**: Using features in Pivotal Tracker the client can specify what requirements they have and then we flesh them out (usually by talking with them) into acceptance criteria, and eventually Real World features.
- **Continous Integration**: We use a CI server I developed called construct. This was built with the idea of being like Integrity, but with background jobs and better support for branches. Now that Integrity has background building, there's still the branching support keeping us "ahead".
- **Code-Coverage Tools**: RCov sometimes. We're not really fussed with code coverage as we test EVERYTHING before we write it. If it exists, there's something that will test it.
- **Communication (Wiki, Mail, IM, Mailinglists, other documents)**: We communicate with our clients using Email primarily, but we also have "standups" with them

using Skype. We've also used Basecamp for this. I'm wanting to use Google Wave for our next project, just as a little experiment. I think it'd be really helpful.

- **Team size**: We have 4 people in our team, usually 2 pairs unless someone's sick.
- **Meetings**: We have a "scrum"/"standup" in the mornings lasting about 15 minutes. The idea of this is that you go over what you did the previous day, any problems you encountered, what you're going to do today and something "new and shiny" you found. This should not turn into a project meeting. Those are for after the standup if required.
- **Bug tracking**: Again we use Pivotal Tracker. The client can file a bug here and then (ideally) write how to duplicate it. Then we write a test to ensure that this should never happen again (aka: Regression Testing) and it goes through the same work process as our features, we deliver and the client accepts.

Share

answered Dec 5, 2009 at 4:37

Ryan Bigg

108k ● 25 ● 241 ● 262

1

vote

I Work for Chillisoft.co.za in South Africa

Test-Driven-Development: We have been using Test Driven Development practices since the first Kent Beck Book it is enforced throughout. We use NUnit and R# as the test runner.

Do you test?: In addition to TDD we do manual testing (Visual) and this is automated where necessary. Technologies used for automation depends on UI Technologies.

Unit Testing: See TDD.

Integration Testing: Yes but we not yet used ubiquitously.

Acceptance Testing: We do custom software development for external customers you don't get paid untill they accept hence yes.

Code Reviews: Scheduled bimonthly for every project. Even those that have been pair/peer programmed.

Pair Progamming: We do most of our coding in pairs but there are certainly some tasks and some stages of the project where this is less efficient. What we do is during project startup (first few weeks of each phase) we pair program. In the finishing stages we do not. We also have specific times (8 hours per week per developer) when we work on open source projects these are all pair programmed. All our machines are setup with multiple keyboards and mouse to facilitate the smooth interaction between devs.

Innovative Technologies: We have done a large amount of work on Habanero and use this framework along with a DI container Unity and RhinoMocks.

Agile: We have been using agile philosophies for 8 years and are continuing to experiment with tools and Philosophies as we continue down this path.

Requirement Specification (How?): We capture user stories (Use Cases) for the next iterations in MSWord. We then capture the summary of these in Jeera with effort estimates etc which manages draw down graphs etc.

Continous Integration: We are currently using Hudson which works on top of SVN.

Code-Coverage Tools: We run code coverage for every project as part of our nightly build. We have intergrated the resulting report into the Hudson reports so that we can track these daily for every project.

Communication (Wiki, Mail, IM, Mailinglists, other documents):Obviously we commmunicate in many different manners we have an internal Wiki etc.

Team size: We have 15 software developers.

Meetings: We have a "scrum" every the mornings lasting about 10 minutes.

Bug tracking: We use different systems for internal bug tracking (i.e. during development and internal testing) and for external bug tracking i.e. bugs from customers. Internal tracking (i.e. during internal testing and dev) we use redmine. External tracking (i.e. for our customers) we use Mantis.

Share

answered Jan 27, 2010 at 13:02

GloryDev
680 • 5 • 9

0
votes

•Test-Driven-Development - Just starting to take over, very happy with it so far

•Do you test? - Of course, everyone does, who wants QA laughing at them?

•Unit Testing - For about 2 years now, has helped with stability and tests are run every build

•Code Reviews - Here and There, especially with any late changes

•Agile - Love Agile and its repsonsiveness

•Pair Programming - Just trying it out in a few spots, early returns promising

•Continous Integration - CruiseControl.NET for the Win!!! Such a huge help

•Code-Coverage Tools - Always during every unit test run, CC.NET publishes this info out to the world

•Communication (Wiki, Mail, IM, Mailinglists, other documents) - WIKI, IM, Campfire

•Team size - small, when a product team gets to big we break down into feature teams

•Meetings - short and not often, more likely to get hallway get togethers

•Code metrics - Only cyclomatic complexity

•Static code analysis - Really trying this More use FxCop and VSTS's homegrown

•Bug tracking - TFS for windows and Traq for Mac

Share

answered Oct 23, 2008 at 12:31

Alex

13.2k ● 3 ● 33 ● 46

0

votes

Testing: I do a lot of system testing, and a far smaller amount of unit testing. I try to use test driven development when it makes sense, but it feels like most of the time it doesn't make sense for the core of what I'm doing.

As for the rest, I'm not sure if I properly do "domain-specific languages" or not, but I do use a lot of automatically generated code to capture the repetitive stuff in my work -- I count 9 Perl scripts generating nearly 100,000 lines of code.

As for the rest, team size is always one. I use PC-Lint for static code analysis about once a year. I use gprof and valgrind quite heavily (you don't seem to have mentioned this class of tools). I've been pining for a proper bug tracker system for years now, but am still using to-do list software and e-mail to handle it at the moment.

Share

answered Oct 23, 2008 at 12:32

Sol

2,185 ● 1 ● 13 ● 9

0

votes

- Test-Driven-Development: very occasionally someone might do it for a component. Also, implementing a public specification which comes with conformance tests offers some of the advantages of TDD, and lots of that goes on.
- Domain-Driven-Design: no
- Model-Driven-Design/Architecture: no
- Do you test?: yes
- Unit Testing: some, although not complete. A lot of components are libraries for customer use. There's a fine line between unit and functional testing of a "strlen" implementation.
- Integration Testing: not really, there's little between unit and system tests
- Acceptance Testing: yes, and subsets of the acceptance tests used as system tests
- Code Reviews: no formal process, but some code gets reviewed
- Innovative Technologies (Spring, Hibernate, Wicket, JSF, WS, REST, ...): no
- Agile: no
- Pair Programming: no

- UML: no
- Domain-specific languages: very occasionally
- Requirement Specification (How?): sort of
- Continous Integration: no, but daily builds and reversion of failure-causing changes at discretion of the test team
- Code-Coverage Tools: no formal requirement, test team have been known to use 'em
- Aenemic Domain Model: I don't even know what this is
- Communication (Wiki, Mail, IM, Mailinglists, other documents): all of them, chosen ad hoc except that requirement and design docs must be HTML under source control, and internal interface documentation is generated from Doxygen comments in headers.
- Effort estimates: a bit
- Team size: about 20 programmers, variously grouped into component teams of 1-4 people. Pretty much nobody works exclusively on the component whose team they belong to.
- Meetings: weekly full meeting to exchange progress reports and otherwise share what's going on. No other regularly scheduled meetings for developers: discussions arranged as required.
- Code metrics: no
- Static code analysis: not unless you count -pedantic ;-)

- Bug tracking: Bugzilla, somewhat integrated with source-control

Share

edited Oct 23, 2008 at 14:06

answered Oct 23, 2008 at 12:30

Steve Jessop

279k ● 40 ● 469 ● 709

0

votes

- Test-Driven-Development: no.
- Domain-Driven-Design: no
- Model-Driven-Design/Architecture: we do start with models for our apps
- Do you test? yes. Sometimes I'm the only person who tests my stuff. I hate this.
- Unit Testing - no. This is a lacking area in my skillset I consider high priority to rememdy.
- Integration Testing - no
- Acceptance Testing - sometimes. Tough to get the users to go through with it, even with threats from On High.
- Code Reviews - no. We have discussed doing it but never do in the end. I'm frustrated about this.
- Innovative Technologies - no
- Agile - we're mildly agile, though not precisely through a pre-meditated effort

- Pair Programming - no
- UML - as little as we need to, but we do model (we're more deliberately agile here).
- Domain-specific languages - no
- Requirement Specification (How?) - we do. My group is sometimes primarily responsible for requirements gathering. We are typically assisted by a Biz Analyst now but that wasn't always so. The Veep sometimes hands us requirements that come from I don't know where. Sometimes they're old things that were never done but had been planned ages ago. typically gathered requirements are placed into a Word document reviewed by the primary users as well as my team, the Biz Analyst, and the Veep.
- Continous Integration - nope
- Code-Coverage Tools - no
- Aenemic Domain Model - I'm not familiar with his, but no.
- Communication (Wiki, Mail, IM, Mailinglists, other documents) - just email and face to face. I broached this subject recently because we need to do something more, a wiki preferrably. It was placed on the back burner.
- Effort estimates - yes, but we don't make any attempt to really track them. This is another area where I am lacking.

- Team size - 3, myself included (Director <- Team Leader <- Me).
- Meetings - we meet once a week, though not always. Boss usually checks in at least a few times a week individually. Larger group meetings take place sporadically. And of course we schedule meetings to hash out project requirements as necessary.
- Code metrics - nope
- Static code analysis - nope
- Bug tracking - we log errors. that's pretty much our bug tracking.

That's it. We have areas I feel like we could improve on.

Update:

We lost our business analyst to a large layoff a couple of weeks after I posted this (early November 08). I've since implemented ELMAH in an existing application *and* a more recently developed one to assist in bug tracking (we also log to a database) and I love it for the ease of use, the features, and the ability to catch exceptions we aren't catching (which is largely unused, but still nice coverage to have). I'm still poking around with Unit Testing on my own - I really need to pick up the pace there (I also want to learn MVC but I mostly poke around with that too).

We're desinging a new application right now, and I'm doing a mock DB schema (which will get some basic diagrams) for 3 of the 6 modules (Team Leader is working on the

other 3). I'm not looking forward to it, since this will be developed in tandem by the 3 of us (2 modules each) using IronSpeed Designer (6.1). There are things IronSpeed will do for me that I like, but it's not the only way to do these things quickly, and it does some things I don't care for.

Nothing else has changed.

Share

edited Jun 18, 2009 at 13:15

answered Oct 23, 2008 at 15:47

peacedog

1,415 ● 20 ● 32

0

votes

My company has jumped on most of the "buzzword" methodologies. Unit Testing, Test Driven Development, Scrum, Agile, Continuous Integration, Code-Coverage analysis, etc. I find we are jumping from product to product as team sizes change with the economy. We shifted from Rally Dev/Scrum to Jira/Agile after plenty of layoffs. We are using Selenium for automated testing, but now looking at Tellenium and Google's WebDriver.

What are we finding? Sites that have passed every test created for it (including load testing), can be incredible inefficient when truly analyzed. After a code performance analysis we were able to cut server resources by 2/3 for one of our sites, and still had better performance. It still passed the same tests too.

Front-end automated testing does not catch positioning issues that a human would notice in seconds. Sure, we could spend a few hours writing tests to check for positioning. But the tests are brittle and have to be rewritten when page layouts change, even just a little. Testing usually just indicates the code works, not how good it is.

I've worked at big and small companies using many different technologies. Including simple "cowboy coding". There were a lot more bugs when we didn't employ planning and testing methodologies, but we moved a lot quicker. We pushed out changes and fixes in hours, not days and week.

Facebook does a "push" every week (Tuesdays). Often enough there are bugs in the latest code push (not enough testing?), but they often do another push by that Thursday or Friday to fix any issues. My guess is Facebook is closer to the "cowboy coding" methodology and it's been working for them.

Share

answered Jun 18, 2009 at 13:44

Brent Baisley
780 ● 4 ● 4

0

votes

Here are my observations:

- Test-Driven-Development : No
- Domain-Driven-Design : Yes
- Model-Driven-Design/Architecture : Yes

- Do you test? : Yes
- Unit Testing : Yes
- Integration Testing : Yes
- Acceptance Testing : Yes
- Code Reviews : No
- Innovative Technologies (Spring, Hibernate, Wicket, JSF, WS, REST, ...) : Yes
- Agile Pair Programming : No
- UML : Yes
- Domain-specific languages : Yes
- Requirement Specification (How?) Yes
- Continous Integration : Yes
- Code-Coverage Tools : No
- Aenemic Domain Model : No (What do we mean by this ?)
- Communication (Wiki, Mail, IM, Mailinglists, other documents) : Wiki, Mail, IM, Mailinglists
- Effort estimates : Yes
- Team size : 2-4 members
- Meetings : Every Monday fix meetings and every other day floating meetings
- Code metrics : Yes
- Static code analysis : No
- Bug tracking : Yes

Share

answered Nov 1, 2009 at 19:17

Rachel
103k ● 119 ● 272 ● 367

0

votes

- Test-Driven-Development - Yes
- Domain-Driven-Design - No
- Model-Driven-Design/Architecture - No
- Do you test? - Yes
- Unit Testing - Yes
- Integration Testing - Yes
- Acceptance Testing - Started
- Code Reviews - No
- Innovative Technologies (Spring, Hibernate, Wicket, JSF, WS, REST, ...) - No?
- Agile - Yes
- Pair Programming - Yes almost all of the time
- UML - Nothing more formal than lines and boxes on whiteboards.
- Domain-specific languages - A little
- Requirement Specification (How?) - No, we try to get user stories if possible
- Continous Integration - Yes
- Code-Coverage Tools - No
- Aenemic Domain Model -

- Communication (Wiki, Mail, IM, Mailinglists, other documents) - Wiki, IM, Email, Word Docs
- Effort estimates - We use a combination of T-Shirt size (S, M, L, XL etc) and a points system for sprint by sprint velocity.
- Team size - 6->8
- Meetings - Daily stand up
- Code metrics - No
- Static code analysis - No
- Bug tracking - Bugzilla / Version One

Share

edited Nov 1, 2009 at 19:19

Moayad Mardini

7,321 ● 5 ● 43 ● 58

answered Oct 23, 2008 at 12:25

Adam Pridmore

164 ● 1 ● 12

0

votes

- Test-Driven-Development - No
- Domain-Driven-Design - No
- Model-Driven-Design/Architecture - No
- Do you test? - Sometimes
- Unit Testing - Almost never
- Integration Testing - Yes
- Acceptance Testing - Sometimes

- Code Reviews - Only occasionally
- Innovative Technologies (Spring, Hibernate, Wicket, JSF, WS, REST, ...) - No
- Agile - No
- Pair Programming - No
- UML - on my marker board, yes.
- Domain-specific languages - C++ is domain specific, right?
- Requirement Specification (How?) - I think we meet 'em.
- Continous Integration - Yes
- Code-Coverage Tools - No
- Aenemic Domain Model - What's a domain model
- Communication (Wiki, Mail, IM, Mailinglists, other documents) - Email & Skype. What's a wiki?
- Effort estimates - 1-2 days for any given task
- Team size - 2 software engineers, 10 hardware engineers
- Meetings - 2 times a week
- Code metrics - No
- Static code analysis - No
- Bug tracking - No

edited Nov 1, 2009 at 19:20

Moayad Mardini
7,321 ● 5 ● 43 ● 58

answered Oct 23, 2008 at 12:24

Nate
19.4k ● 27 ● 73 ● 93

0

votes

- Test-Driven-Development - Nope, on purpose.
- Domain-Driven-Design - Nope, we're still figuring out the domain.
- Model-Driven-Design/Architecture - Nope
- Do you test? - We test the builds, and get power-users to test.
- Unit Testing - Nothing formal (no nUnit etc)
- Integration Testing - No
- Acceptance Testing - Yes
- Code Reviews - Occassionally.
- Innovative Technologies - random SharePoint tools
- Agile - Yes
- Pair Programming - No
- UML - Never
- Domain-specific languages - Nope
- Requirement Specification (How?) - We stay light on this and iterate. We have a BA that does some requirements analysis, but usually we just invite the

customer to our planning and daily meetings as we go. No formal docs.

- Continous Integration - Yes (cruisecontrol.net)
- Code-Coverage Tools - No (but we do use Visual Studio Code Analysis)
- Communication - Outlook
- Effort estimates - ballpark, double it, then double it again.
- Team size - 2-4
- Meetings - everyday at 9am (scrum!) plus a weekly planning/review meeting
- Code metrics - Nope
- Bug tracking - Bugzilla
- **Source Control - SVN**

Share

edited Nov 1, 2009 at 19:21

Moayad Mardini
7,321 ● 5 ● 43 ● 58

answered Jun 18, 2009 at 19:38

Andrew Lewis
5,256 ● 1 ● 28 ● 32

0

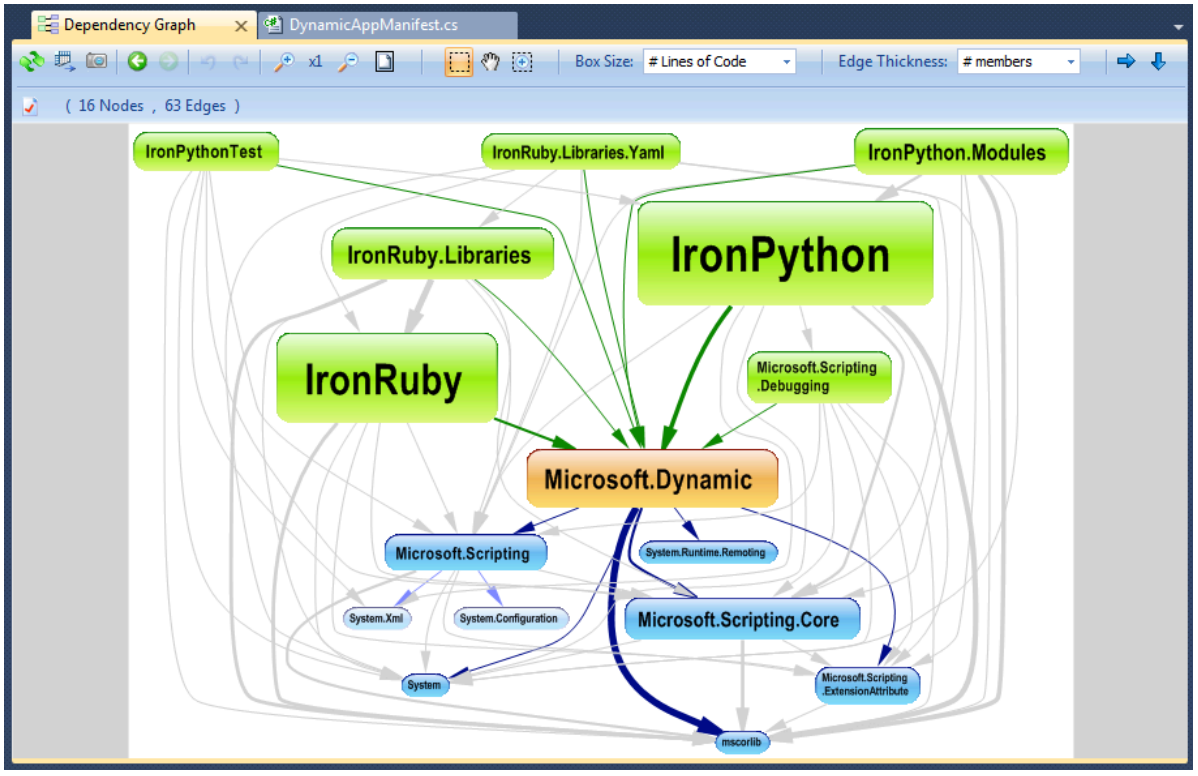
votes

Did you have a look at NDepend? The tool analyze C# and .NET code and comes with plenty of cool features to browse the analysis results. With NDepend you can write

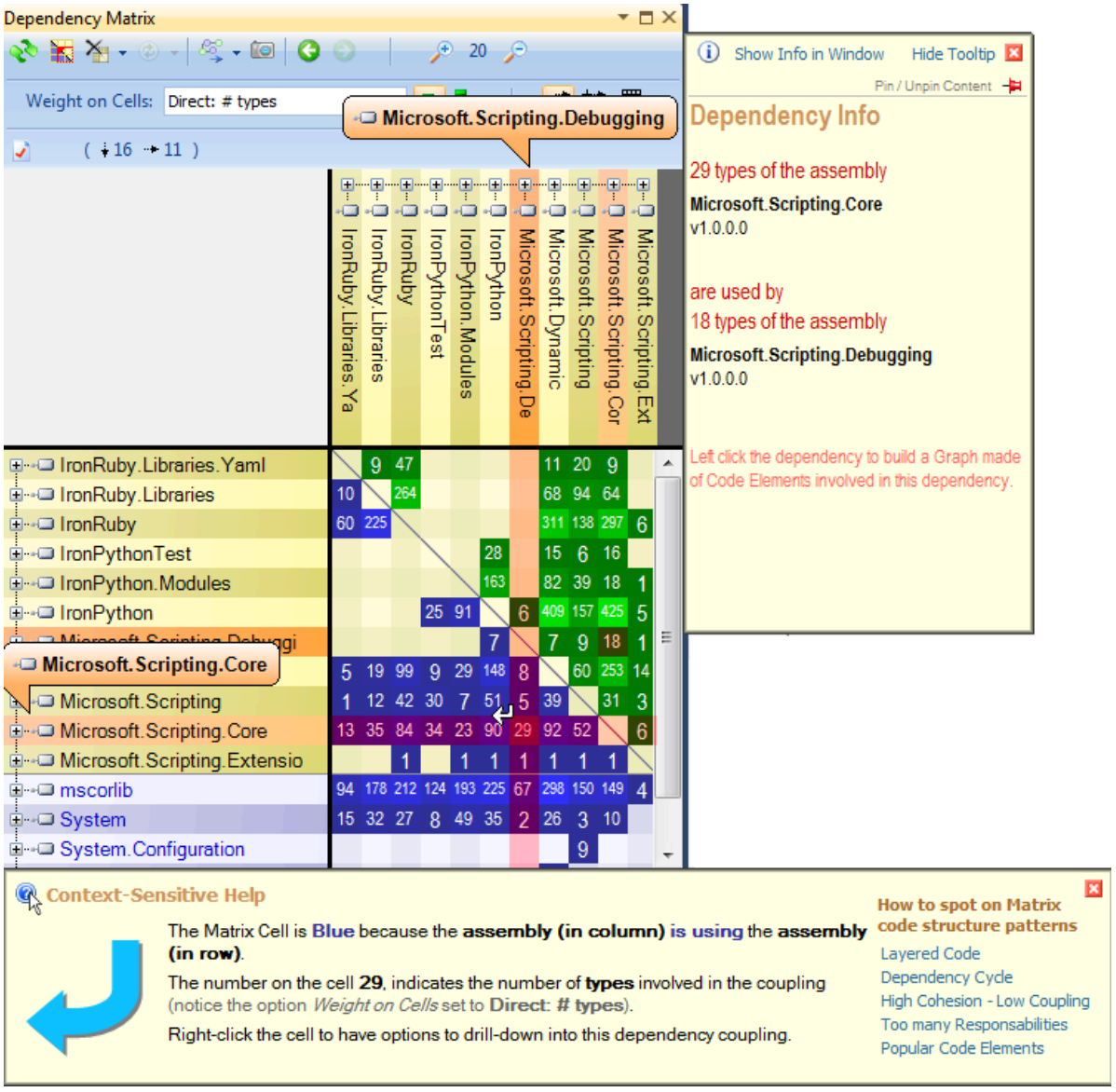
rules on code, you can compare 2 versions of the code base and you can harness more than 80 code metrics.

Also, the tool comes with several great visualization features like:

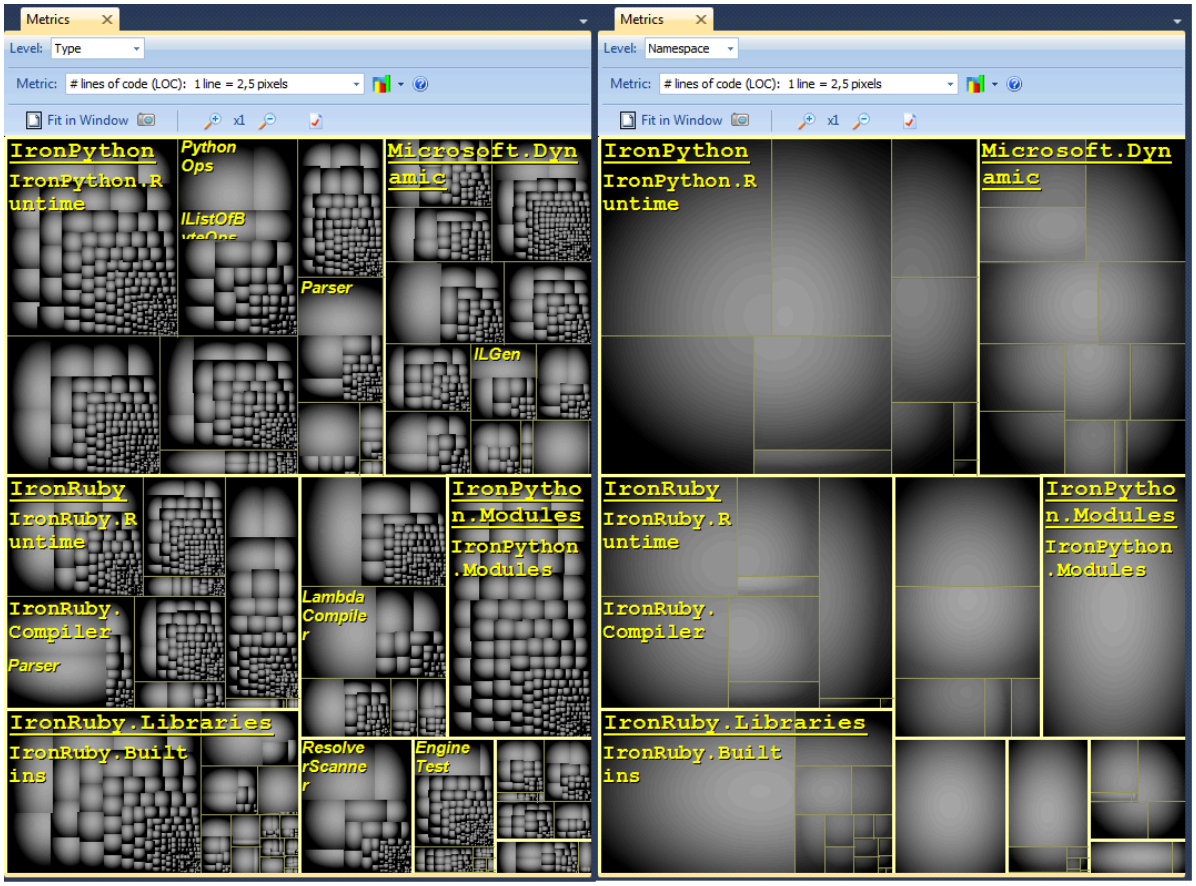
Dependency Graph:



Dependency Matrix:



Code metric visualization through treemaping:



Share

answered Oct 18, 2010 at 18:09

Patrick from NDepend team

13.8k ● 6 ● 67 ● 106

-1

votes

its nice to hear that MDA, DDD and Pair Programming is not used anywhere :D Martin Fowler is not god, just guys with some weird ideas.

- Test-Driven-Development - if you want to
- Unit Testing - yes
- Code Reviews - kindda
- Innovative Technologies (Spring, Hibernate, Wicket, JSF, WS, REST, ...) - yes, Seam

- UML - kindda
- Continous Integration - yes and no
- Code-Coverage Tools - yes and no

Share

edited Oct 23, 2008 at 13:54

Steve Jessop

279k ● 40 ● 469 ● 709

answered Oct 23, 2008 at 12:49

mark

2 There are weird people who does not even try to learn something, not ideas. – Arnis Lapsa Jun 18, 2009 at 13:53