Computer Science 370 Software Engineering

Interview 1

Owen Smallwood, developer at Coconut Software

Favorite programming language: Python

U of S graduate favorite class: Computer Science 306

Dog or cat person: Dog

Engineering department- mainly product people and developers

Product- high level road map and short

Developer- How we make that and reach goals

Sprints every two weeks

Every team, one product person, decided priorities for sprint and what they want to build

Developer tasks those features

Process? Agile

What are you working on day to day?

Generally, each pick off top, product manager prioritizes them top to bottom, some tasks other developers know better than other, so might pick farther down, if bug comes, that gets higher priority

If working on story, don’t know what to do, what do you do?

If don’t know how should function, talk to product to understand better. E.g. client wanted one thing, product came up with very overengineered system

If technical, just ask team

What is your tool set looking for versional control, communication, etc.?

* Use Bitbucket
* Use Jira

Is everyone in company in Discord?

* Main company communication Slack
* Now, Discord, can have rooms, really just developers

A lot of product team, when giving requirements, how technical are they very high level? E.g. User needs to do or database

* Depends on product manager
* Some have computer science degrees, some developers in past, others haven’t
* For ones not very technical, go over it at tech time, say we want to this, how long would it take, get an estimate

Have there been any hard lessons or things you weren’t prepared for?

* Biggest thing learned going from school, when student, under impression, to be good software developer, only had to be a good programmer, just small fraction of being good software developer, definitely may soft skills that will make life easier as a software developer
* Situations to avoid, have brought down app many times, tools in place to make it very apparent when that happens, don’t want culture afraid to make mistakes

Tell me about releasing on Friday.

* Gone to release thing on evening, or Friday, broken, ready to go home, something goes wrong on weekend, their responsibility to clean it up
* Tuesday and Thursday, now just deploy

What sort of peer review exists? Working on story, finish code, what happens?

* Make a pull request, teamer would review it
* Would tag, at least team lead
* Two approvals before can merge into development branch, which gets deployed
* Functionally test features, TDT?
* Some things pretty don’t have to test
* UI, have to test

What level of testing happening?

* Various feature test, some unit tests
* Test suite takes 6, 7 minutes to run
* Tried end to end but incredibly slow but doesn’t really scale
* Trying to make something that runs at night, smoke test

What does continuous integration look like?

* Use AWS and Kuper what
* Whenever push feature branch, circle ci runs test suite
* Step
* Builds containers, if any of that fails, get email saying why
* Finiker to deploy, QA deom interation

What is last final advice?

* Working in a group project, nothing like real life, if someone dos nothing, not much you can do, it’s good, learn to deal with conflict
* Also liked, no source of right answers, sometimes things messy, nowhere to go that says this is right answer
* Kind of has out amongst selves

Interview 2

What do you do at 7Shifts?

Tell us a little about what you do.

* Helped scale and grow engineering team to just over 50
* Hired brand new chief technical officer
* Right now, managing technical direction and technical vision of infrastructure team, made up of a couple of developers, entire sole purpose to be enabler of other teams
* Lay architecture foundation or command-line shortcuts for them, can focus on developing product more quickly

What’s your favorite language? PHP

How many manhole covers in Saskatoon? 67000?

Speaking of development processes, how does 7Shifts actually make things? -7Shifts is a Sask product, software as a service, means hosted and run in Cloud, would access through browser where we specifically (don’t know word here, homeblax?)

* Not bound by older school traditional software products, product that they control and host and can operate at own pace, own speed
* Subscription model, pay them monthly for continued shipped improvements in product
* Work on standard agile framework
* Operate in two week sprints, in which individual team sits with product owner, makes priority list for what company should deliver next for that team, get the tickets, prioritize, plan them out, acceptance criteria, get designer from designer, start working on products, set goal for sprints, so for next week need to at least deliver xyz
* Other companies, usually wait until end of two weeks sprint cycle to deliver product, they don’t do that, if ticket done, ready for customer, they deliver it right away, typical day usually 15-20 updates, minor bugs improvements, performance improvement, brand new features
* 2 week sprint cycles, prioritization by product team, write code, test, and deliver as quickly as can

How often do you ship?

* Shipping multiple times a day instead of 2 months

How do you make sure things not going wrong when shipping so much?

* Need to have a lot of confidence in code base, that what you’re touching, changing or adding to doesn’t break behavior of what customer’s already used to do
* Easiest way, large automated test suite, there’s a couple, three parts, both to back end code (and front end code?)
* Unit test suites individual suites of code, makes up like 85%, always layered in integration test suite, makes up like 13%, end to end, like 2%, can take anywhere from 5 minutes to an hour to run, once all green, code good, ship it, that’s from automated testing perspective
* Before feature released to customer, still some other aspect, does it meet acceptance criteria, will it confuse the customer, does flow and everything make sense
* Mix of big automation to gain confidence in automatic deployments manual testing that we understand feature and how it works

With testing comes peer reviews and what not, what peer review processes do you have to make sure software good?

* There’s a couple of things you try and automate, also manual again as well
* When developer finishes feature or something, open pull request, use GitHub, as soon as open that pull request, automated test suite starts running against that new code
* What we then ask developers to do for two reasons, one for knowledge sharing, also use for logical errors that might exist, developers do review of code change, rule that needs two people to check it out, now other people know what done and what’s going on accountable
* In pull requests, focus very little on code style, have automated linkers that runs against code checking that matches architectural style of code, standard enforced automatically
* Pull reviews purely focused on discussions on does this make sense, logical sense, will it solve the problem

There’s a level high how a system’s going to work, how much guidance given to a developer?

* Design elements of system usually comes into play during sprint plan, when team presented with problem, get together in room, couple of hours, half a day or whole day, up front, agree how going to break down technical problem, product manager or product owner usually come up with user story “as a user, I need to do xyz”, and then up to development team break down what technical requirements may look like, in order to complete that story, need to store that in this database, need to stream video, need to whatever that might look like, all that stuff is planned as good as possible up front so consensus on team and goal to shoot for, if try to do willy nilly and don’t plan ahead, will end up in chaos
* Stepdad ingrained 7 ps of British military, prior preparation and planning prevent piss poor performance
* Prior preparation and planning maximizes performance
* More that talk upfront, talk is cheap, changing code is expensive

From industry, if you can think of being student moving into industry, what is the biggest lesson you remember learning or the difference?

* One thing, biggest switch from academia into industry life is collaboration and communication e.g. homework, kind of discouraged to talk to other people, don’t want to be seen as cheating or whatever that might look like, if don’t snap out of it quickly, will hold you back, when industry hire you as junior developer, know exactly where you’re at in terms of scale or proficiency, really important to say I don’t know, shows humble enough to admit it, also shows value company’s time, if spend 3 hours then ask for help, perfect trait, if spend 2 days that would’ve taken another 2 or 3 hours because missing some context, not really responsible
* Really important to be able to leverage team to learn more quickly, say I don’t how to do with this, can you pair with me for a second, really important trait as transition from student life into industry life

How important do you think grades are to any company looking to hire?

* Really tough question, personally when interviewing someone, don’t look at grades
* Think really important, how self-motivated to learn things when time is right, if find gap in knowledge, learn about it, contribute it back to team, much more important
* Do care about what classes people took, if know went through 370 and 371, know probably had exposure to group project, shared accountability, should know how to communicate or do testing during development, and concepts they’ve been exposed to, even if not great at it, understand where they come from and they’ve gone through
* Much more focused on how do they communicate, what base line classes have you taken

Advice, how to students finish project working on and not want to kill each other and actually ship soomething?

* One thing everyone learns is software developers notoriously bad at estimating, think better than they are and can get through more
* Regular syncs important
* If working on team project, figure out way to communicate goals, Slack channel, daily updates, what you’re working to get done
* Shared accountability, as a team at end of week, should be able to accomplish x, shared accountability

Interview 3

Who are you? What do you do? Where do you work?

* Shane Giroux, currently COO of Push Interactions, a local mobile software development company
* Right now, about 75% of time, server development with Python and Django

What is your true favorite language?

* Soft spot for C, but now Python is the one for him

What is your favorite beverage?

* Coke Zero

Can you explain Push’s development process? How you go from an idea to shipping?

* Service based company
* Clients go to them, maybe spark of idea or fully fleshed out idea
* Initial meetings, explaining general idea, maybe ultimate goal of idea of software
* From there, got a product manager as well as a designer
* Kind of start working with client from there, talk a lot, set up some wireframes, diagramming out basic flow
* As move forward, vision comes a little more to life, ask do you feel comfortable moving forward with this
* Estimate what it’ll take to build this, break it down into general user stories, pretty high level at first, break it down and we’ll estimate how long each story might take
* From that point on, client gets proposal, usually sign off proposal and start building
* That’s when enter development cycle, developers have all estimates so general idea of software
* When start building, realize whole bunch of questions didn’t realize didn’t know
* Fire off questions, usually takes a day or two and then starts building stuff
* May not get answers for a little while, start building as best as can
* Every week or so, show progress to client, maybe just quick touch base meeting, questions, this far in, might not be able to show for a couple of weeks
* Try to communicate best as can
* As deliver, maybe new things come up or misinterpreted something
* Think finished it, maybe about 90% done at point which leads to polish up final 10%
* Much quicker cycle at the point
* Test it out, say we want instead of this, tweak it a bit until final push off to store

How many apps do you make?

* Usually a dozen apps a year, some bigger, some smaller
* Some clients, really niche idea, e.g. just a converter, only takes about a week, bigger clients federated Co-op maybe 3, 4, or 5 months to finish up

You mention stories, what software methodology follow and how closely follow?

* Kanban approach, where not breaking things up into strict sprints
* Going to try moving that into soon, past few times, didn’t seem to work due to client availability e.g. two weeks and client not available, hard to plan next sprint
* Go kanban, try and prioritize as best as can, build in this order makes the most sense, as developers finish off stories, just grab next one that’s highest priority

You mention estimation, how do you estimate and how do you get better at estimating?

* Estimate base on ideal days, if we knew everything about this story, it should take a couple of stories
* Generally rely on own experience and developers
* Don’t have big formal system where we can look at okay, we did this in the past, log in of this type, estimated 2 days, took 5 days
* Estimate on past experience, estimated 3 months, took 5 months, think what happened

What tools are you using and how has this changed now that remote?

* Moved to Jira instead of physical borad
* Jira good have stuff in Cloud, have backup
* Probably aren’t using to full extent, used to use Stash, Git Stash (not actually sure what he said)

What sort of checks do you have involved during process to make sure doing right thing?

* Every time story finished, goes through pull request process, always tag two developers and one QA person (PA?)
* Developer says here’s my story, try and explain best can how it should work, QA person also has general broad idea, toss it to them, say built this, client asked to build this
* Might not give them everything and see if they get the flow, might think something client might not have came up with
* QA involved every step of the way, encourage them to think of flows client or them didn’t think of

Do your developers typically work on QA one project or bouncing on multiple projects?

* Lots of bouncing
* Prioritize based on which client is being the squeakiest wheel, who did you promise the most, who’s paying the most
* Will do as much juggling can for developers, don’t have to think about all the context switching, so even if just for day can focus on project
* Try to let them work on for one or two weeks so can get really into that flow

As mobile, what do you think of native development compared to cross platform development and what you use and why?

* Strictly native shop
* Have employees or new hires that say why don’t try xyz
* Earlier many cross functional frameworks, now kind of, merging together into one or two big ones
* Next year, may look at one of those
* Reason stuck with native is generally faster on device, tended to keep up with newer OS releases
* Back in Android breaking releases quite often, a lot of the frameworks not keeping up in time and how do you explain to you client, your app’s going to break are not work for 6 months until your framework is ready

What do you do in your polished phase or other phases when you realize you might be going over budget or how do you bid on that? Is it fixed or is there wiggle room?

* Can depend, some clients are more willing to have wiggle room than others
* Some clients, personality, or actual budget, can’t spend more than this
* Sometimes end up in conflict, because don’t want to put software for client that doesn’t meet their quality standards
* Are times agree with client that it needs it, have to take a bit of hit, toss it in, gesture of good will
* Most part, have to have really honest conversation where client says oh, thought it would do this and say wasn’t in specs, proposal do say this is an estimate, anything beyond that will be based on time approved
* Sometimes, have to look inwardly, our client asked for log in system, says log in all done, client says where’s forgot password feature, say that’s fair enough, didn’t say explicitly included
* Terminated clients before when they didn’t put in extra work, given extra to clients when more flexible, bit of a give and take

Do you have industry lessons on something going poorly and rebounding and changing perspective?

* Became much more chill on how things go
* We will make mistakes, client will make mistakes
* Database backups, got to have those, make big schema changes, backup your database before you roll it out, check that actually exist, not just can restore from them, but are they still there or not
* Put yourself in client’s shoes
* Deploys are terrifying thing for him, with put some code, immediate deployment
* Should be super careful, won’t ruin database, app crash, log people out
* Testing it over here, migrating it over here
* Deploy is a very crucial step

Do you have any advice for students in project groups, how can they work with multiple people in multiple roles?

* Communication actually more important than coding skills when it comes to working in the group
* Honest up-front conversations, say don’t do this, this will happen
* Other people realize okay to say
* Work on communication, learn how to communicate with other people
* Everybody communicates in different way, may say something that you take offense to, but you shouldn’t, try to be very chill about it, say code broken, say okay, let’s take a look at it
* Communication at root of every business problem, better at communicating, better at developing software

Interview 4

Tell us who are you. What do you do?

* Doug Johnson
* University of Saskatchewan, graduated 2005
* Went to (Vicema?) Networks
* Now work in the office of the CTO, principle software architect
* Role is around technology strategy, new product initiation, picking out what cogs we’re going to work on
* Also, standards work, standards for ca
* University on research projects

What class is your first level 60 character?

* Warrior

In your standards and in documentation, are you doing UML in industry and what do those standards look alike?

* Kind of, a subset of UML is definitely used, but not full book
* Basics, yes, really good communication tool, provides mechanism to look at picture and build mental model of what you’re talking about, so either somewhat strict UML or loose UML to get on same mental model, definitely internally at Vicema and in standard bodies

What can you tell me about the development process, from taking project to building and getting out the door?

* Multiple layers to that
* Vicema sells to cable industry
* Do video, IPTV, or linear qualm broadcast, also do dockless internet, cable mode (what .\_.), some analytics platforms
* Kind of two components, hardware aspect, have long lead times to get hardware, high price point of entry, can’t just open bin and produce hardware, lot of steps to it
* Waterfall part of a product, or hardware part of product, very close to waterfall, figure out everything that product will need to do, look at available hardware chips that will help us do that, come up with an architecture of those hardware blocks, some fixed, need to run sensor so need this microchip, a lot of them more flexible (a confusing example), get board, get it printed, put it in (something), developers can start accessing
* Then, kind of before that once have hardware, software part is agile, because once have hardware booting and entire space of possibilities in hardware, to make any hardware work, or any possibilities work is software, so software is iterative and agile, constantly doing releases, constantly working on these features, turn these possibilities into a product
* Kind of two layers, that hardware part is very waterfall, dated, process heavy, expensive, and then software much lighter weight process, hardware constrains what is possible, software enables those possibilities