**Nearsoft Academy**

**Week #4**

**Video Reviews (12)**

**Cesar Dario Garcia Jiménez (Nearsoft Academy)**

[**cgarcia@nearsoft.com**](mailto:cgarcia@nearsoft.com)

**1st Video**

# **“Programming the Universe”**

Author: Seth Lloyds

(2012)

url: <https://www.youtube.com/watch?v=I47TcQmYyo4>

Programming the universe is a lecture from the professor Seth Lloyd to explain how the universe is computing data since the beginning of it existence, everything including us is the result of this computing, and now we are trying to understand what is going on over those processes and how can we interact (perhaps to say how can we hack) with the information of the universe that is already there.

Personal Thoughts:

So, programming the universe?yes.

Scientists and Physics are now understanding (in part at least) how particles behave and react to their stimulus in the most microscopic world we know so far that is the Quantum World, that means that we can program the littlest pieces of physical matter (densities of energy) to store or process information in the way we want, even while it behaves in a weird way to understand.11

The thing is that we are not “creating” those behaviors in the matter, matter itself is the result of the “universe program” that starts running in the beginning of our known time, and is up to us how are we going to use this information and what are we going to do with it.

By the way...It is funny to hear Sith Lord (Seth Lloyd) talk about the power of controlling energy densities...coincidence? I don't think so!...

(It's a joke).

**2nd Video**

# **“****Quantum Machine Learning ”**

Author: Seth Lloyd

(2014)

url: <https://www.youtube.com/watch?v=wkBPp9UovVU>

What I've learned on this:

The information presented here is very interesting although it is not easy to digest because I have not formally studied Quantum mechanics, still I do understand in the large picture many of the concepts introduced.

Quantum Mechanics in relation to machine learning presents such a powerful concept tool because on one side machine learning allow us to work with huge amounts of data to create self-improving software and added to the power of QM calculation speed it's something that opens a lot of possibilities to all sorts of models, its application in medicine, weather, cryptography, economics, etc. is just amazing, so let's see what we and others can bring on this topic.

In short what I learned in here is that since both, QM and Machine Learning works by calculating vectors, you map and migrate your algorithms from classic computation to Quantum computation.

**3rd Video**

# **“****Richard Feynman, The Great Explainer: Great Minds”**

Author: SciShow

(2013)

url: <https://www.youtube.com/watch?v=JIJw3OLB9sI>

I loved this video, it is a short description of Richard Feynman, one of the most important mathematicians of the 20th century.

The curious thing about his description is that he was not only a remarkable mathematician, but he also was a whole personality, he was something you normally won't expect from a science man, as the description explains he regularly do math on strip clubs and play bongos there as well, he used to practice jokes over his fellows in places that one normally would not even think to do and this is inspiring because it is a reminder to not judge anyone by their appearance or personality.

**4th Video**

# **“Computing a theory of everything ”**

Author: Stephen Wolfram

(2010)

url: <https://www.youtube.com/watch?v=60P7717-XOQ>

This presentation is extremely important and interesting as well because it deals with one of the main goals of computation that is to systematize knowledge and Stephen Wolfram is taking this to the next level, ambitioning to create the most powerful general-knowledge engine.

Learnings:

Stephen Wolfram is doing an inspiring job showing us the possibilities we have in the computational universe, for example the creation of software engines that are able to receive a human natural language as input and produce the expected formal knowledge data as output, the implications of this is that this brings another level of democracy over computable knowledge since all the people will be able to communicate with systems and compute data in an effective way no matter their language or technical background, they will only need to ask in a natural way what they want to know.a

He also show us a principle named “The principle of computational equivalence” which says that “incredible simple systems can do computations as sophisticated than everything”, this comes from his experiments making simple software to work in different patterns to prove how many different and sophisticated results he can get, as if simple software computational results were like the initial cells of the game of life, they eventually create the most sophisticated structures.

There is more on the video to explain but I find it difficult to explain it myself because I haven't fully understand how, but he show us how a simple computational model can produce tridimensional models that represents universes and that you do not need to get very sophisticated on this to get universes that expose gravitational properties or other forces that exist in our known reality, and this get us closer to create models that allow us to understand our universe.

In his words: “Things with simple roles can have serious implications about the limits of science, about predictability, controllability, about the intelligence of the universe, about questions like free will and about creation technologies”.

I would investigate more over these kinds of work because I am personally interested in contributing to the way we all access to world history files and how everybody access knowledge.

**5th Video**

# **“****Feynman and Computation”**

Author: Tony Hey

(2011)

url: <https://www.youtube.com/watch?v=9miKIWIYi4w>

In here Tony Hey gives us a short presentation about Feynman and his contributions not only to physics but to computing as well.

Feynman was a genius understanding and explaining nature in the physics, mathematics and when the digital systems were born as computers he predicted that we wouldn't be able to compute nature using classic computers because they are limited by electric laws, and some nature phenomena like the ones that occurred in the Quantum Worl works under different laws, so he states that we would need quantum computers to achieve a complete study of nature.

He designed the foundations for parallel processing computing called pipeline which is when the output of one process is the input of another one.

I agree with Feynman on the need of Quantum computers to experiment and calculate phenomenon that occurred on the quantum world or in the abstract world in like mathematics and many others, still there is a lot of work to do over this fields because right now is still too expensive and too complex to build, experiment and understand those phenomenon without being a physicist or a mathematician, perhaps classic computing improve more and more into its interface for delivering the knowledge and be able to learn more efficiently and take us one step forward in understanding and improving our current tools for understanding nature.

**6th Video**

# **“****Reminiscing about Richard Feynman ”**

Author: Danny Hillis

(2011)

url: <https://www.youtube.com/watch?v=8CKW4A6jnJA>

This is a short lecture about how Danny Hills first meet Richar Feynman and one of the latest talks they had before Richard died.

This is something to reflect about dead.

Richard Feynman was a great contributor to science but still he couldn't do anything about his cancer and still when Danny tell him “I am sad because I know that you are going to die”, and Richard calmly replied “yeah...that bugs me sometimes too...but, it's actually not as bad as you think...” and he continues explaining why and says: by the time you are at my age and told as many stories as I do, you will realize that, even when I'm dead, I will not be completely gone...

Even though we know we are temporal organisms, we have always dreamed about being immortals, somehow the desire for surviving over time has been in our DNA since the beginning of mankind and Richard knew that he had left a mark that was going to remain over time.

Personally thinking of death is emotionally very uncomfortable to me, I have a little son that depends on me and the simple fact of thinking of the death of me or him gets me really nervous, I don't want to die yet.

**7th Video**

# **“****Feynman on Scientific Method”**

Author: Richard Feynman

(date unknown)

url: <https://www.youtube.com/watch?v=EYPapE-3FRw>

Very interesting footage from a class imparted by Richard Feynman in which he talks about the scientific method.

Many people talk about the personality of Richar Feynman but look a footage of him talking really gets you an idea of how he thinks.

Learnings:

Richard exposes his skeptical position to everything unless it is proved under scientific rigor.

We have to keep in mind that even when we create experiments to prove certain theory, we still don't understand nature in the deepest way and this always leave us with a certain margin open for error even when some hypothesis is proved to be right or wrong, because as we keep understanding more and how nature really works, we are able to develop better experiments and Richard jokes about the tendency in science when a theory is proved right, later on is proved to be wrong, later on is proved to be incomplete, and later on is proved to be right again but with some changes as we create better and better experiments each time...

**8th Video**

# **“****Tools for Continuous Integration at Google Scale ”**

Author: John Micco

(2012)

url: <https://www.youtube.com/watch?v=KH2_sB1A6lA>

John Micco explains to us how google implements his continuous integration system to continually be testing every piece of code delivered in a concurrent way providing a trustful platform that ensures the quality of their software.

In here the expositor discusses some of the strengths and weakness of their system for continuous integration, for example:

Learnings:

Continuous Integration is a development practice to integrate code into a common repository several times a day, and then each check-in is verified by an automated tool allowing teams to detect problems early.

Google implements this practice but with a difference, the classical cycle of continuous integration run tests in a linear way while changes in the code come in, this makes it very difficult to find and fix localized problems.

i.e. Classical Continuous Integration Cycle.

------> Linear check-ins & test runnings -------> Fails at the end of cycle.

|1st commit 2nd commit 3rd commit

| | | |

|1st test start... |2nd test start … |3rd test start...|FAIL

Google concurrent implementation of Continuous Integration cycle.

-----> Parallel check-ins & tests runnings ---> Fail 1st commit but no 2nd.

1st commit

| |

|1st test start... |2nd test start … |3rd test start...|FAIL

|

2nd commit

| ||1st test start... |2nd test start … |3rd test start...|Succeded

This allows them to localize problems more easily because this way we know that the 1st commit fails the test and 2nd one succeeded.

So in summary:

**Google Continuous Integration Process:**

-Tests on every change submitted.

-Fine-grained dependency.

**Benefits**: **Costs**:

-Identification of failures sooner. | - Enormous investments in

-Identification of culprit changes precisely. | compute resources needed

| incrementally as teams and

| projects grow.

*This allows developing more Safely but at a very high cost.*

Another important thing is the Flaky Tests, this test doesn't allow to accept or reject software changes in a very precise way.

Google measure the rate of success or failures of every test run, so they can focus more in the tests that have unnormal rates, this way it is easy to detect flaky tests.

**9th Video**

# **“****The Release Process for Google's Chrome for iOS”**

Author: Ivan Ho & Lindsay Pasricha

(2014)

url: <https://www.youtube.com/watch?v=p9bEc6oC6vw>

The release process for Google's chrome in IOS is a story full of obstacles in which Google had to manage their way to get their browser as an option for IOS users.

Learnings:

Linda Pasricha tells us how their adventure on developing a version of Google's chrome for IOS was.

She explain to us that, it is very different to develop a version of a desktop web browser for mobile device because of the difference of resources available in both kind of devices, so, how do you bring the same experience on mobile?, when they first develop Google's chrome for mobile was a native application of their own OS for mobiles, in this story the things were different, Apple compete with Google in the mobile market so naturally they will cooperate with caution.

The first problem was to adapt the architecture to a system where the app can not have the same resources available because it is not native in IOS.

The google chrome team start to become stressed out because they had to work faster to deliver features on time because now they had to sum the time that the Apple store take for approval, besides their processes for quality are different so they have to conceal this, it was not easy, but they achieved in getting Chrome into IOS with success.

Testing is part of the engineering culture inside Google and they consider bigger than just a matter of QA, testing it is becoming a burgeoning field in computer science; Testing is a subset of engineering productivity.

Testing should be the result of a collective effort, the team must find a balance between quality and the velocity that takes delivering a new feature, this is achieved by the optimization during building, developing, testing and the release phase.

Testability: Google tries to create code in a way that is easy to test, perhaps they need to refactor some code to make it easier to test.

They implements release automation, testing automation but some manual tests are also required, load and stress testing, and more, all inside their Continuous Integration cycle.

Lindsay Pasricha Google has a set of channels managed by bots to do automatic tests over those channels, they run unit testings, performance testings, and screenshots testings, this channels exists to test software of different phases (beta, experimental, releases, etc), in addition sometimes it is needed to run manual tests when some automated test isn't implemented yet or when some technical debt is acquired.

**10th Video**

# **“****I Don't Test Often ... But When I Do, I Test in Production”**

Author: Gareth Bowles

(2014)

url: https://www.youtube.com/watch?v=xkP70Zhhix4

Gareth Bowles works on a team at Netflix called Engineering Tools and he and his team are responsible for providing the build and deployment automation.

This time, Gareth talks about the Testing culture over Netflix and it is something very impressive.

Learnings:

Another very important thing to note in here are the details exposed about how the teams at Netflix work which are also something really cool and interesting and it deserves a second look to understand what we can learn from them.

Netflix runs over a Distributed Systems architecture and it has a big infrastructure since their service consumed by people represent approximately the 34% of the internet traffic (according to Gareth Bowles).  
  
So the first point is **Testing**:

*1) Building distributed systems is hard.  
2) Testing them exhaustively is even harder.*

Gareth explains to us that the classic testing methods were short for covering the size and complexity of the systems, and they have learned from previous disasters suffered by the company that they need to work on testing using a different approach.

The goal of the testings for Netflix is the Availability of their service.

So what about the story of the disasters that Netflix suffered and how is this related to their Testing process, well...

Since Netflix is a service consumed by a large percentage of people around the world, it strongly depends on the availability of their service that is distributed in the Amazon AWS services & Datacenters, they need to have several instances of their services running to ensure the permanency of the service when one or more instances are down, if a lot of those instances go down, the service may not be able to maintain itself available to cover the demand for their service.

They have suffered problems where entire regions lost availability of their service due to unexpected failures of their instances, Amazon data centers suffered an electric problem one time, and Netflix wasn't prepared for this kind of situation, another problem was when an engineer accidentally delete several instances and again Netflix wasn't fully prepared for this situation.

What does this brings to Netflix learnings:

Failures in production lead them to practice Production Code Coverage tests, this was something really good because:

Testing in production allows you to use Real Time-Usage Patterns under testing and from this you can evaluate your Tests, they use tools like “Cobertura” which allows them to measure the number of time a line of code was executed to find bottlenecks and performance issues and to see if there is any dead code to remove.11

To improve their testing process Netflix they implemented the “Chaos testing technique”, which they recommend a lot, they constantly launch automatized “chaos monkeys” to simulate failures or stress the system to the maximum to test it out and see if they can hold on effectively those tests.

Point number 2 is **Workflow & Teamwork**

Netflix culture for working is very interesting and something to learn from too.

Gareth describes this culture with the words “**Freedom & Responsibility**” which they describe as: “Each development team that makes the couple of hundred services of Netflix to run, can set their own schedule, they can deploy as frequently as they like and at whatever time they like—***as long as they work with other teams to make sure dependencies aren't impacted obviously.11***

They manage their own capacity and auto-scaling in terms of – **so they are responsible for knowing the load and their service and react11 accordingly**”.

This kind of liberty incentives the teams to *design* to avoid failure from the beginning and this is one of the reasons why they test more in production.

This kind of liberty and responsibility is great but because people make it great, I know we at Nearsoft work under the same principle which is something new for me, so I hope I can learn from this philosophy and be a useful element for the team, the company and the world in general.

**11th Video**

# **“****Test coverage at Google”**

Author: Andrei Chirila

(2014)

url: <https://www.youtube.com/watch?v=4bublRBCLVQ>

This is a short lecture about the importance of the coverage of tests in software and how is Google doing at this.

Learnings:

The test coverage is important because it is the way to ensure that new things don't broke what you already have and the dependencies you have must not be affected.

Test Coverage at Google starts with:

**Code Reviews** (At google it gets done as):

* Reviews before check-ins.
* Web Based.
  + Streamlined.
  + Small, incremental changes.
* Mandatory.
  + Enforced by tools.
* Automated analysis.
  + Fully automatized.
  + Style Guide compliance
  + Do the tests pass?
  + Fast

**Types of Test Coverage:**

* Function (Was a particular function called?)
* Statement (Was a particular line of code executed?)
* Branch (Was an edge in the program executed?)
* Other (Increasingly expensive to calculate...)

Test coverage is optional at Google so not all the projects that are under development implements Test Coverage, the percentage of projects that do implements it is the 85%, but Google does aim to achieve the test coverage for 99% of the projects under dev.

What does this mean?

Google has collected coverage related data to ensure champion code and they have developed an optional system to apply Test coverage to their projects.

**12th Video**

# **“****The Testing User Experience”**

Author: Alex Eagle

(2014)

url: <https://www.youtube.com/watch?v=J7c0Bw840X8>

Alex Eagle from Google explain to us what testing means for the Google Cloud Platform today, the platform captures all of the builds and test results from almost everything that runs on Google.

So, What happens next after we get all the test results, and what do we do about failures?

Engineers are responsible for writing the code and fixing the bugs of their code, but for many be testing their code may feel like a waste of time, it is not always easy to convince them to fix code that is partially or completely broken when they are working on something else, even when they know the benefits of testing.

The speaker throw a nice methaphor, it's like telling people to eat healthy, they want to do it, they know its good... but they don't like to be told to do so, and often people don't eat healthy all the time...

Learnings:

Of course, Testing is necessary and it represents a benefit for engineers and their projects.

It is important to change the way we feel about testing, there's where the title of this presentation comes.

Tests do represent a cost, and like any other process it has to be administrated inside the software development cycle, it depends on the circumstances of the project and the organization and it may or may not be convenient to do full coverage testings depending on the requirements of the project, time, money, and criticality of the software.

Tests should always be done but sometimes they can become a trade-off in certain situations, despite that, tests are necessary to ensure the quality of the products so, do it always when possible.

But why it keeps feeling like a waste of time or an extra effort that can be skipped sometimes?

The user experience provided by the existent tools for testing are not as friendly as many would want, having to see huge log files for error checkings aren't the best experience ever, Google implemented something that improved the involvement of the engineers with the testing exercise.

Introducing the “Breakage”:

Google realize that engineers had a better response when the tests that produce failure results comes with a suggestion and a more accurate description of the problem, They introduced static analysis into the toolchain of developers and the tool produce suggested fixed to the most common bugs.

Don't report the “broken window”, but give tools to fix it!.

(from the Broken Window theory).

In summary:

When you are able to do a good model of a problem it's easier to repair it, so finding a good way to model and present the problems in software is a good way to pull engineers more into good testing practices.

# **Personal Summary from the Assignment of Week 4:**

# As to the pretotype assignment:

# The pretotyping excersice is great, definetely it is something to develop and practice more because it allows us to test and find solutions or improvementsin a better and faster way that we can implement in our lives.

Computer & Science:

This is one of the most interesting aspects because it take us to the root of what lies underneath computation, we still have work to do in order to understand nature and we must open our mind to the world of posibilities we still have, systemizing knowled is one of the most interesting topic I found in here, I have a personal interest in it because technoloy is allowing us to develop tools that can change the way we transmit information and there is were the collective knowledge relies, if we want to change things for the better we need to be ables to transmit and receive information in better ways, I hope I can do work related to this in the future.

Testing & Automation:

This topic may not sound so interesting at first, but after giving it a try it turn out to be truly an adventure where you have to use your reason, critical thinking and ingenuity to test software always with a scientific and a pragmatic approach to ensure that you are actually developing something valuable, what is the point of creating something that is not valuable, that is has no quality in it...testing is an excersice that must be done at pair with developing software, even though I do not know much about the topic, I found it usefull and interesting.

Extra:

The model for working in Netflix is both very interesting and awesome, I write my thoughts about it above in the lecture review.