

# Hönnun og smíði hugbúnaðar

Software architecture and implementation

Haust 2022



# Ég

- Tengiliðaupplýsingar

- Þórður Friðriksson
- 8441537
- [thordurf@ru.is](mailto:thordurf@ru.is)
- Piazza (Frekar en email / sími)

- Nám

- Bsc í hugbúnaðarverkfræði við Háskólann í Reykjavík - 2020


- Núverandi starf

- Hugbúnaðarsérfræðingur / Solution Architect fyrir Motus



# Kennsluhættir / teaching methods

- Fyrirlestrar

- 2x 45 min
- Teknir upp á Echo360
- 08:30 á þriðjudögum í stofu V201
- 14:20 á föstudögum í stofu M201
- Kenndir á íslensku (glærur og verkefni til á ísl. og ensk.)
- Gestafyrirlestur í lokin frá  SYNDIS um öryggi

- Dæmatímar

- Unnið í skilaverkefnum
- Dæmatímakennarar:
  - {Óákveðið}
- **Hópur 1** -> 10:10 á þriðjudögum í stofu M106
- **Hópur 2** -> 11:50 á þriðjudögum í stofu M108
- **Hópur 3** -> 13:30 á þriðjudögum í stofu M108

- Piazza

- Aðal samskiptamiðill
- Piazza > Email
- Bónus stig fyrir nemendur sem eru duglegir að svara



# Námsmat



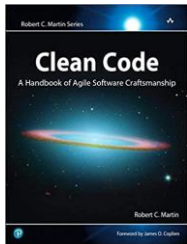
- Lokapróf
  - **30% af lokaeinkun**
- Lab verkefni
  - **40% af lokaeinkun**
  - 10 verkefni í heildina
  - Lægstu tvær einkunnir detta út
  - Late submission policy:
    - Nemendur fá samtals 96 klst(4 dagar) sem auka buffer fyrir sein skil á lab verkefnunum
    - Þessi buffer má dreifa yfir lab verkefni að vild
    - t.d. ef skil á lab6 er 24.09 þá mætti skila 28.09 án refsingar eða ef lab2, lab3, lab4, lab5 á að skila 27.08, 03.09, 10.09, 17.09 respectively þá mætti skila þeim 28.08, 04.09, 11.09, 18.09 án refsingar
    - Þegar þessi buffer er búinn þá dregst einn heill af verkefna einkunn fyrir hvern dag sem líður
- Projects
  - **30% af lokaeinkun**
  - 2 verkefni í heildina
  - Late submission policy:
    - Nemendur fá samtals 96 klst(4 dagar) sem auka buffer fyrir sein skil á project-unum
    - Þessi buffer má dreifa yfir project-in að vild
    - Þegar þessi buffer er búinn þá dregst einn heill af verkefna einkunn fyrir hvern dag sem líðu
- Piazza bónusstig
  - Hægt er að vinna sér inn bónusstig með því að vera með mörg góð svör/spurningar á piazza
  - „ef ég gef þér x upvote á piazza þá máttu búast við y upphækun á lokaeinkun“.
- Bónus stig í skilaverkefnum
  - bónus stig í project-um og lab verkefnum eiga bara við um þau verkefni og eru bara ætluð til að hækka verkefna einkuninna sjálfa
  - T.d. ef þú færð 107 stig í Project 1 út af bónusstigum þá hefur þessi auka 7 stig ekki áhrif á lokaeinkun heldur telst verkefnið bara sem 100

**Ath. Submission Policy-in** fyrir lab verkefni og Project-in eru aðskilin þ.e.a.s þið hafið 96 tíma buffer fyrir lab verkefni og síðan hafi þið líka 96 tíma buffer fyrir project-in, ekki er hægt að nýta buffer-inn fyrir lab verkefni í project-unum og vice versa.

# Námsefni

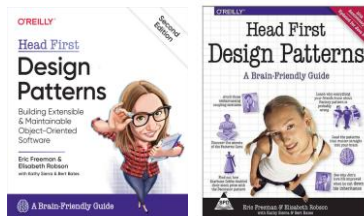
# Bækur og heimildir

- Clean Code

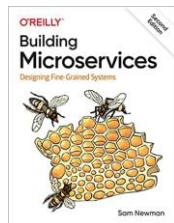


- Head First Design Patterns

- 2nd edition en 1st virkar líka



- Building Microservices



- Aðrar bækur

- Óþarfi að kaupa
- verður flokka sem „auka lestrarefni“
- Mun reyna að hafa nægilega ýtarlegar glærur/glósur/fyrirlestra úr þessum efnum



- Greinar

- Mun linka á greinar
- Sumar skyldulesning
- Sumar auka lestrarefni (Mun raða í röð eftir mikilvægi)

# Hvað er Software Architecture

- Engin góð skilgreining
  - 10 mismunandi svör við google
  - Skilgreining er umdeild
  - Margar skilgreiningar úreltar
- Nokkrar skilgreiningar
  - Eitthvað sem lýsir stærstu þáttum kerfisins
    - „High Level“
    - Vandamál -> „hight level“ er mjög afstætt
  - Sameiginlegur skilningur
    - Þessi sameiginlegi skilningur á kerfinu er architecture-inn
    - Inniheldur hvernig kerfið er skipt upp í einingar og hvernig þær tala saman
  - ákvarðanir sem þurfa að vera gerðar snemma í ferlinu
    - „*fundamental structural choices*“ sem er dýrt að breyta
    - Vandamál -> architecture þarf ekki endilega að vera eitthvað sem er erfitt að (t.d. Microservices)
    - Vandamál -> "*we made good decisions, but only now do we understand how we should have built it*"

# Uppáhalds skilgreiningin mín



**“Architecture is about the important stuff. Whatever that is”**



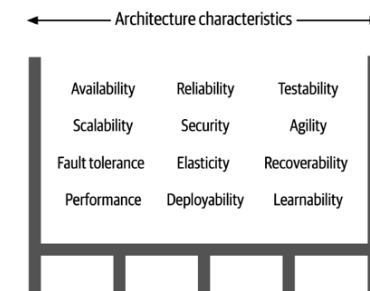
# Architecture is about the important stuff

- Góður arkitekt er góður í að þekkja og finna mikilvægu hlutina í kerfinu
- Góður arkitekt er góður í að þekkja þá **þætti sem eru líklegir til að vera með vandamál** ef það er ekki lögð nægileg hugsun í þá.
- Það sem er “*mikilvægt*” er ekki eins í öllum kerfum
  - T.d. forritunarmál, mikilvægt í sumum kerfum en ekki öðrum

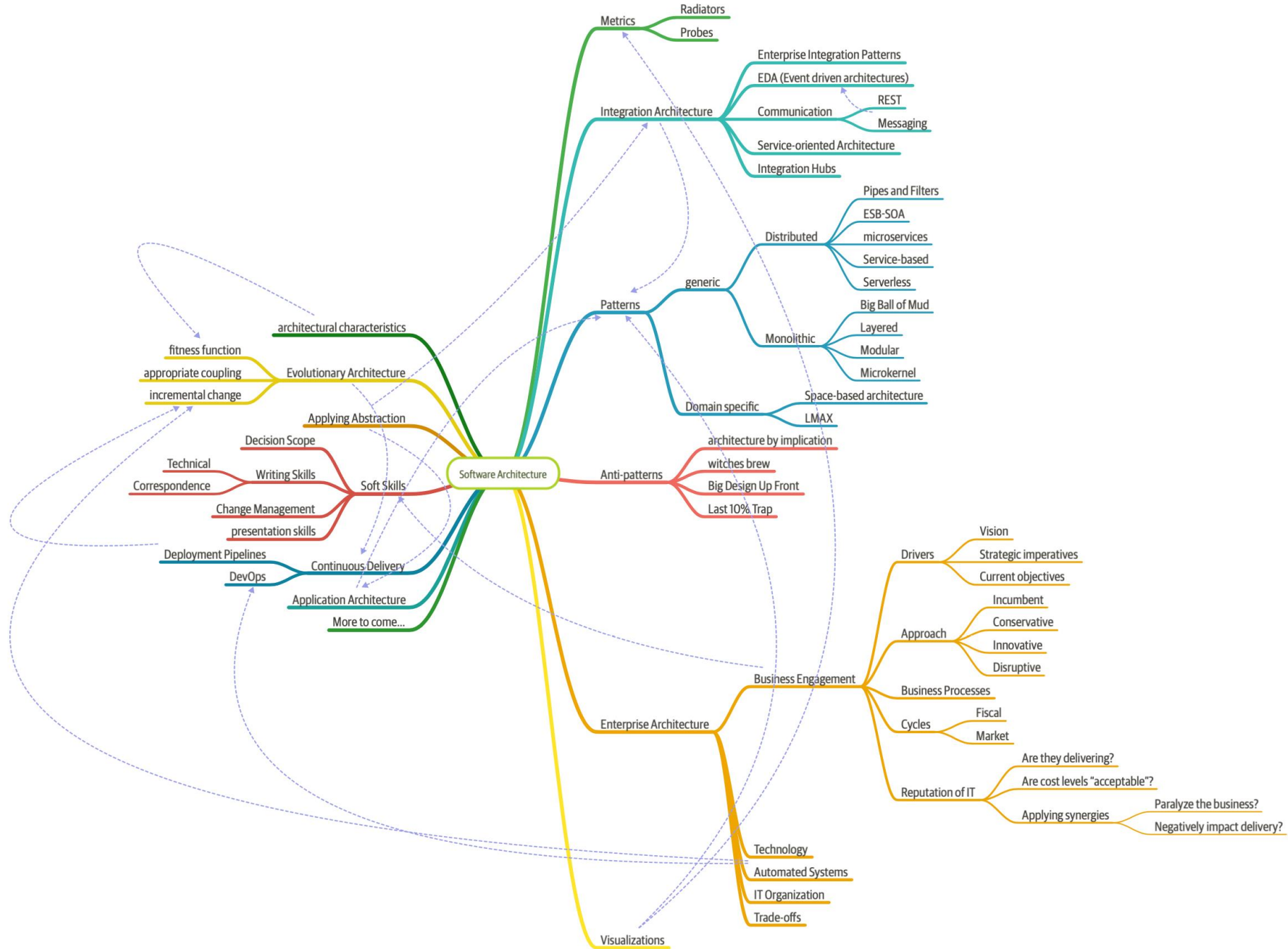
# Dæmi um architectural þætti

- Hvernig kóði er skiptur í module-a(umdeilt, [sjá glæru](#))
- Deployment strategy
- Design patterns and principles(umdeilt, [sjá glæru](#)):
  - Abstract factory pattern
  - Dependency injection
  - Builder pattern
  - Etc
- Tradeoffs
  - Scalability vs Performance
  - Cost vs development speed
  - Etc.
  - „Programmers know the benefits of everything and the trade-offs of nothing. Architects need to understand both. – Rich Hickey
- Architecture styles eins og
  - Microservice
  - Monolith
  - Event-driven architecture
  - Etc.
- Technology choices eins og
  - Forritunarmál
  - Deployment platform
  - Gagnagrunnur
  - Version control (átti meira við í gamla daga, git er standardinn í dag)
  - Etc.
- Framework choices eins og
  - ORM frameworks (Nhibernate vs Entity framework vs Dapper)
  - Web frameworks (Vue vs Svelte vs React vs Angular)
  - Etc.

- Greina business og domain requirements
- Hugsa um „ Architecture characteristics“



- Architectural rules(harðfastar reglur) eins og:
  - Presentation layer-ið má ekki tala við persistence layer-ið
- Architectural guidelines(ekki harðfastar reglur) eins og
  - Nota async messaging á milli service-a til að auka performance
- Programming Paradigms
  - Procedural
  - Functional
  - Object Oriented
  - Logical
  - Constraint
  - Etc



# Fleiri skilgreiningar

- *„Architecture is that which is fundamental to understanding a system in its environment”*
- *“Architecture is a set of architectural design decisions”*
- *“Architecture are things that people perceive as hard to change”*
- *“Software architecture refers to the process of translating software characteristics into a structured solution that matches business and technical requirements. What are the software characteristics? They’re traits such as security, flexibility, scalability, or reusability.”*
- *“Architecture consists of the structure combined with architecture characteristics (“-ilities”, i.e. scalability etc.), architecture decisions(i.e. prevent the presentation layer from calling the persistence layer), and design principles (guidelines that aren’t hard set rules i.e.async messages in microservices)”*
- *“For me, software architecture is simply the combination of application architecture and system architecture, again in relation to structure and vision. In other words, it’s anything and everything related to the design of a software system; from the structure of the code and understanding how the whole works”*
- *“The architecture is the primary carrier of system qualities such as performance, modifiability, and security, none of which can be achieved without a unifying architectural vision. “*
- *“Architecture is the stuff you can’t google”*

# Samband forritara og arkitekts

- Ættu arkitektar ekki að forrita?
- Ættu forritarar að vera arkitektar?



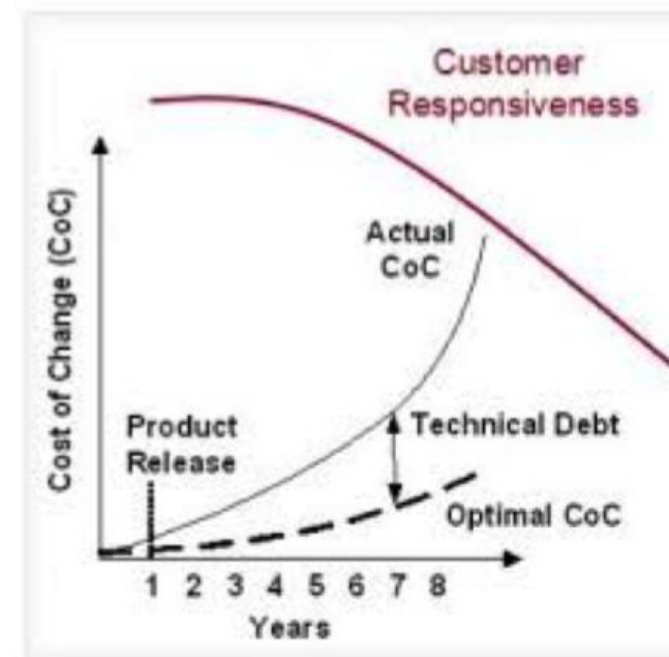
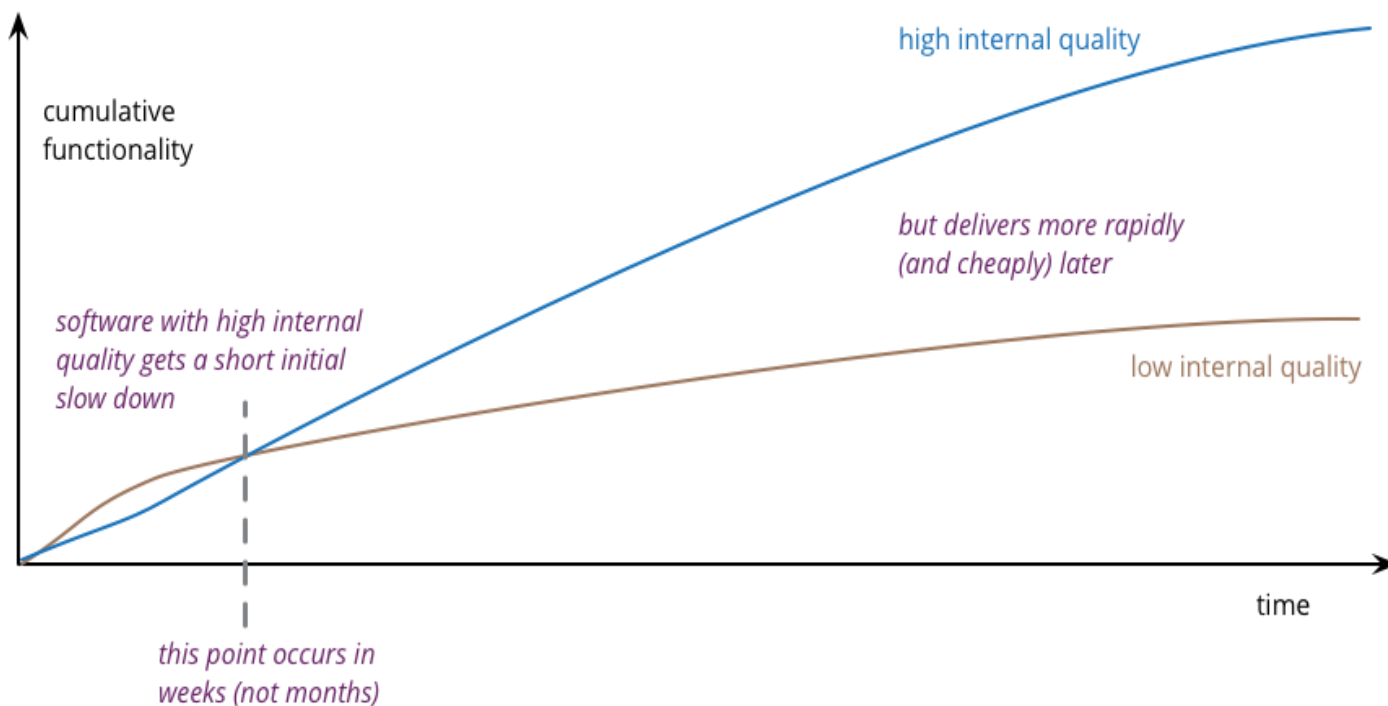
# Það sem er búist við af arkitekt

- Hugsa um og ákveða/guide-a Architectural þáttunum (sjá [glæru](#))
- Stöðugt greina arkitektúrinn
  - Koma með tillögur að bætingum.
  - Annars er hætta á *structural decay*
- Fylgjast með nýjustu tækni
  - *technical breadth* mikilvægara en *technical depth*
  - “A jack of all trades is a master of none, but oftentimes better than a master of one.”
- Fylgjast með hvort hönnunar ákvörðunum sé framfylgt
- Hafa reynslu
  - Þarft reynslu til að taka réttar ákvarðanir
  - T.d. reynslu við ýmiss tækni, framework, umhverfi etc.
  - *technical breadth* mikilvægt fyrir arkitekt
- Þekkja business-inn og domain-ið
  - Erfitt að taka réttar ákvarðanir án þess að þekkar þarfir kerfisins
  - Arkitekt þarf að geta skilið domain-ið til að ákveða architectural þætti (t.d. scalability, performance, availability etc.)

# Af hverju að læra þessa fræði



- Slæmur arkitektúr
  - leiðir til *big ball of mud*
    - Ólæsilieki
    - Breytingar erfiðar
    - Breytingar hættulegar
    - Virkar kannski í byrjun en ekki til lengdar
      - „attention to internal quality pays off in weeks not months.“ – Martin Fowler
  - Leiðir til *technical debt*
    - Speglar tímann sem það tekur að útfæra eitthvað útaf slæmri hönnun
    - Geta verið litlar ákvarðanir sem safnast saman, “Ég laga það seinna”
  - Leiðir til þess að fólk með reynslu verður dýrmætt
  - Leiðir til þess að það verður erfitt fyrir nýtt starfsfólk að koma sér inn í kerfið
  - Leiðir til þess að fólk hættir að vilja vinna í kerfinu
- Skiljanlegur og læsilegur kóði er king
  - Eyðum miklum tíma í að skilja og lesa kóða
  - Minnkar líkur á villum
  - Gerir breytingar auðveldari



# Big Ball of Mud



*A Big Ball of Mud is a haphazardly structured, sprawling, sloppy, duct-tape-and-baling-wire, spaghetti-code jungle. These systems show unmistakable signs of unregulated growth, and repeated, expedient repair. Information is shared promiscuously among distant elements of the system, often to the point where nearly all the important information becomes global or duplicated.*

*The overall structure of the system may never have been well defined.*

*If it was, it may have eroded beyond recognition. Programmers with a shred of architectural sensibility shun these quagmires. Only those who are unconcerned about architecture, and, perhaps, are comfortable with the inertia of the day-to-day chore of patching the holes in these failing dikes, are content to work on such systems.*

—Brian Foote and Joseph Yoder

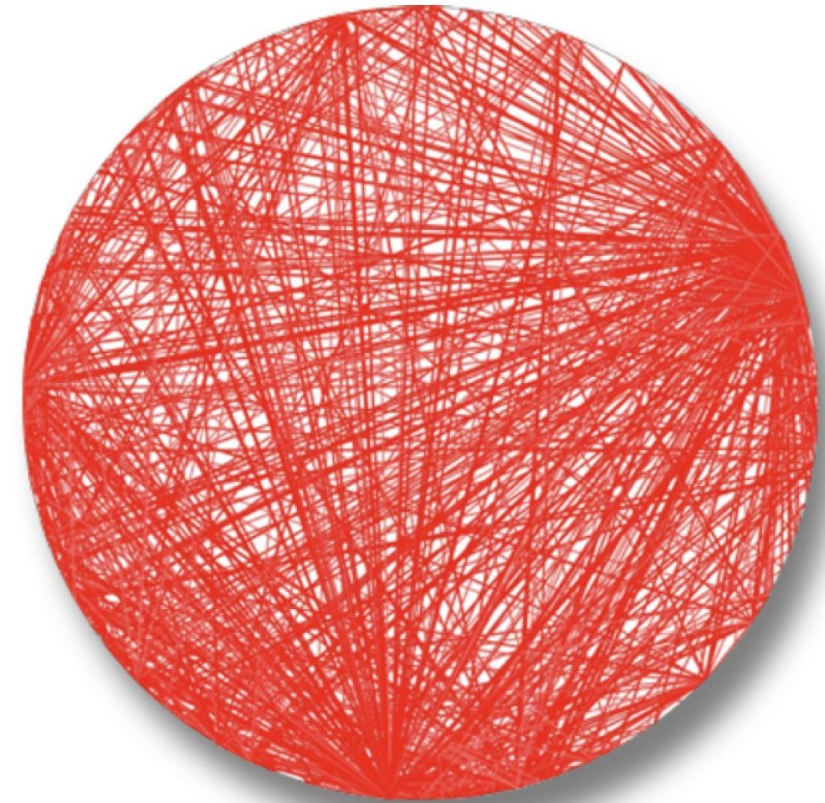


Figure 9-1. A Big Ball of Mud architecture visualized from a real code base

Hver punktur á jaðri hringsins er klasi og  
hver lína er tenging milli klasa

# Learning outcomes

- Kunna að beyta og velja hönnunarmynstur þar sem við á
- Læra um clean code og design principals
- Læra um mismunandi architecture styles og vita hvenær á að velja hvað.
- Læra að smíða Testable hugbúnað
- Læra á Docker
- Læra um aðra Architectural þætti(sjá [glæru](#))
  - T.d. scalability, security, logging etc.



# Hvað þessi áfangi er ekki

- Þessi áfangi er ekki Greining og hönnun hugbúnaðar
  - Ekki UI/UX
  - Ekki requirement analysis eða user analysis
  - Ekki UML
  - Etc.
- Þessi áfangi er ekki Hugbúnaðarfræði
  - Þessi áfangi og þessi fræði deilir vissulega eitthverjum/mörgum sameiginlegum þáttum með Hugbúnaðarfræði en ég mun ekki taka hluti fyrir eins og
    - Agile
    - Scrum
    - User stories
    - Estimation og planning
    - Git
    - CI/CD
    - Etc.

# Verkefnin

- Lab verkefni
  - Lab verkefnin eru í 10 talsins og eru til skila í næstum því hverri viku
  - Lab verkefnin eru lítil og tækla þá hluti sem eru að gerast í fyrirlestrunum hverju sinni
  - Lab verkefnin geta bæði verið fræðileg verkefni sem og forritunar verkefni
  - Hægt að vinna í tveggja manna hópum
- Projects
  - Project-in eru 2 talsins
  - Project 1 tæklar aðallega hönnunarmynstur sem við skoðum djúpt í fyrri hluta áfangans
  - Project 2 tæklar aðallega Architecture Styles sem við skoðum í seinni hluta áfangans
  - Project-in eru stærri en lab verkefnin og því mikilvægt að byrja á þeim tímanlega
  - Hægt að vinna í fjöruggra manna hópum
- Reglur um svindl
  - Ég minni einnig á reglur um verkefnavinnu og lokaprófa, ef upp kemst um svindl í verkefnum fá allir þátttakendur 0 fyrir verkefnið sem um er talað og ef upp kemst um svindl í lokaprófi má búast við fall-i í áfanganum og mögulegum brottrekstri úr skóla
- Sjá [námsmat](#) um Late Submission Policy og vægi verkefna

# Tækni og umhverfi

- Forritunarmál
  - Python
- Forritunar paradigms
  - OOP
- Annað
  - Docker

Allt feedback, gott sem slæmt, vel þegið

# Spurningar?

