

# God digitalisering?

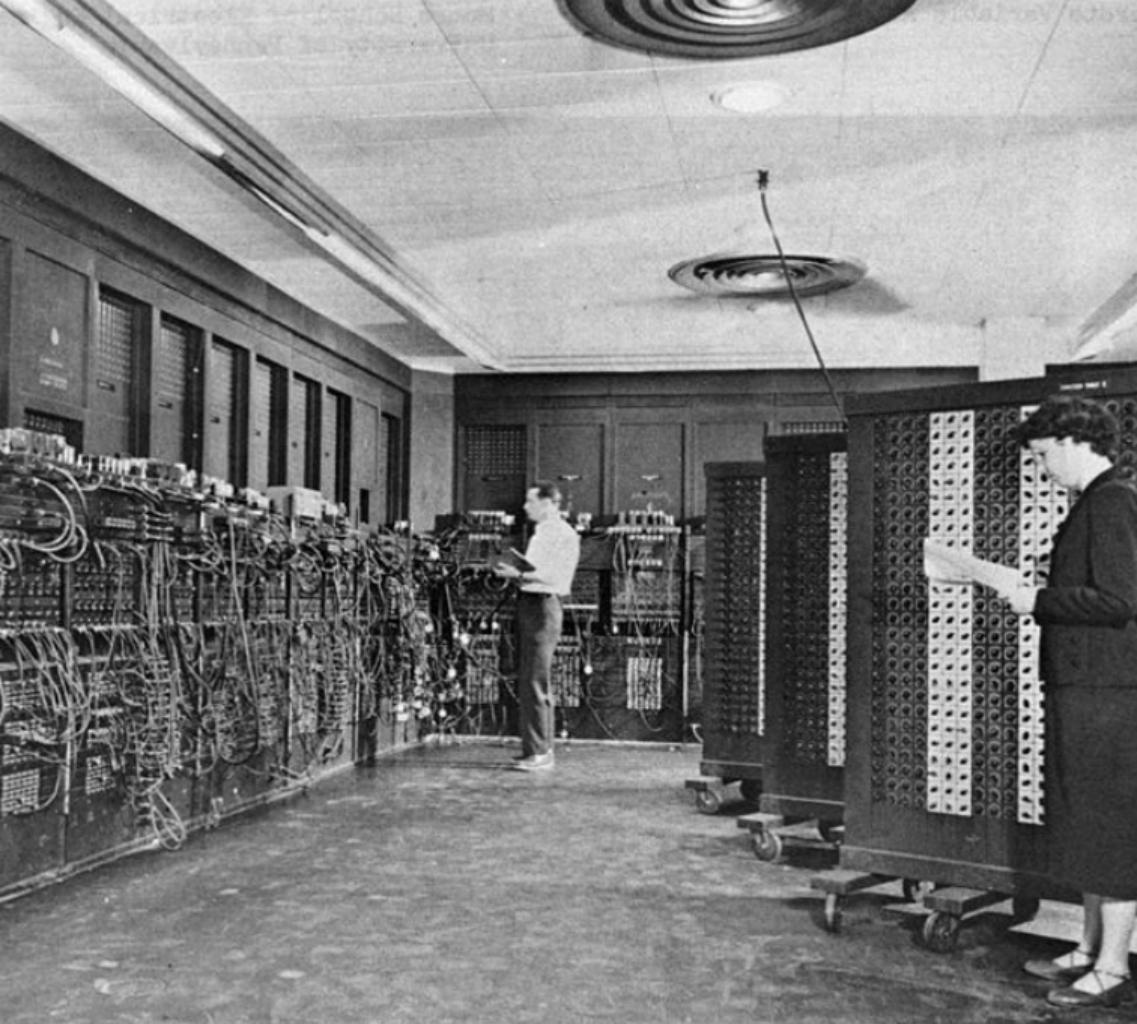
Kravhantering & öppenkällkod för en bättre värld

Björn Regnell

Professor i programvarusystem  
Datavetenskap, LTH, Lunds universitet

# Tekniksprång

- Domesticering 10000 år sedan
- Mekanisering 500 år sedan
- Elektrifiering 250 år sedan
- Datorisering 50 år sedan
- Digitalisering 25 år sedan







**Att skapa  
gemensam  
kunskap om  
framtiden**

# Systemutveckling

krav  $\Rightarrow$  implementation  $\Rightarrow$  drift

# Systemutveckling

krav  $\Leftrightarrow$  implementation  $\Leftrightarrow$  drift

○ kontinuerlig leverans

# Kravhantering = kollektivt kunskapsbyggande

- Det räcker inte att kunna koda...

# Kravhantering = kollektivt kunskapsbyggande

- Det räcker inte att kunna koda...
- Vi måste också tänka ut **vad** vi vill koda och avgöra om det är **rimligt** och **bra** att koda det vi vill!

# Kravhantering = kollektivt kunskapsbyggande

- Det räcker inte att kunna koda...
- Vi måste också tänka ut **vad** vi vill koda och avgöra om det är **rimligt** och **bra** att koda det vi vill!
- Vi skapar kunskap om framtidens system medan vi bygger dem.
- Många kompetenser behövs: vi bygger vidare på varandras kunskaper.

Valfri kurs i årskurs 4:

# Kravhantering 7,5p

<https://cs.lth.se/krav>

# Vad behöver vi göra?

Kravhanteringens **sammanvävda**  
**grunduppgifter** pågår **ständigt**:

- Elicitering lära
- Specificering modellera
- Validering kolla
- Selektion besluta

# Vad behöver vi kunskap om?

# Kravhanteringens **sammanvävda** **kunskapsområden** utvecklas **ständigt**

# Hur ser sammanhanget ut?

Den **komplexa kontexten** utvecklas  
**ständigt**:

- Intressenter                         användare, makthavare
- Vår produkt                         avgränsning
- Andra system                         samverkan
- Gränssnitt                             interaktion, protokoll

# Vilka är våra intentioner?

Olika **sammanvävda** förutsättningar utvecklas **ständigt**:

- Mål intressebalans
- Prioriteter urval
- Risker skada
- Åtagande resurser

# Vilka produktkrav behövs?

Olika **sammanvävda** kravmodeller  
utvecklas **ständigt**:

- Funktionalitet effekt
- Data tillstånd
- Kvalitet nytta
- Testfall mätbara kriterier

## När leverera resultat?

**Stegvisa resultat levereras kontinuerligt:**

- Road-map strategi
  - Resurser mänskliga, monetära
  - Begränsningar realism, villkor
  - Releaser tid, rum

# En checklista för ditt projekt:

## context – who



stakeholders incl. human users	our product
other systems	interfaces and protocols

## intentions – why



goals	priorities
risks	commitments

## requirements – what



functionality	data
quality	tests

## delivery – when



road-map and strategy	resources
constraints	release plan

<https://github.com/lunduniversity/reqeng/blob/master/reqtbox/reqtbox.pdf>

# Vad är allra viktigast?

- Systemavgränsning                  ⇒ kontextdiagram
- Förstå varför!  
Intressentanalys, målanalys, ...
- Förstå kvalitetskraven!  
Användbarhet, säkerhet, prestanda, ...
- Tekniskt möjligt givet begränsningar? OSS?

Gemensam  
kunskap som  
öppen källkod



**Öppen källkod = våra digitala allmänningar**

# Vad är öppen källkod?

Källkod som

- är fritt **tillgänglig**,
- får **modifieras**,
- får **distribueras**,

utvecklas i samverkande **gemenskap**  
enligt medföljande **licens**.

*Open Source Software (OSS)*

[https://en.wikipedia.org/wiki/Open\\_source](https://en.wikipedia.org/wiki/Open_source)

# Vad är en OSS-licens?

Juridisk text, reglerar användningen.

Två principiellt olika typer:

- tillåtande **permissive**  
*exempel: MIT*
- måste även dela förbättringar **copyleft**  
*exempel: GPL*

Påverkar affärsmodellen, *exempel: neo4j*

[https://en.wikipedia.org/wiki/Open-source\\_license](https://en.wikipedia.org/wiki/Open-source_license)

[https://en.wikipedia.org/wiki/Business\\_models\\_for\\_open-source\\_software](https://en.wikipedia.org/wiki/Business_models_for_open-source_software)

# Öppen eller fri eller stängd?

Det finns lång, intressant historia:

- Free (Libre) software:  
politiskt orienterad, "mänsklig rättighet"
- Open source software:  
kommersiellt orienterad, "ekosystem"
- Motståndare:  
Steve Ballmer, tidigare VD Microsoft:  
"öppen källkod är cancer & kommunism"

[https://en.wikipedia.org/wiki/History\\_of\\_free\\_and\\_open-source\\_software](https://en.wikipedia.org/wiki/History_of_free_and_open-source_software)

# OSS förändrar planeten

Ranking	Project	Leading company	Market Value
1	Linux	Red Hat	\$16 billion
2	Git	GitHub	\$2 billion
3	MySQL	Oracle	\$1.87 billion
4	Node.js	NodeSource	?
5	Docker	Docker	\$1 billion
6	Hadoop	Cloudera	\$3 billion
7	Elasticsearch	Elastic	\$700 million
8	Spark	Databricks	\$513 million
9	MongoDB	MongoDB	\$1.57 billion
10	Selenium	Sauce Labs	\$470 million

Battery Open Source Software Index (BOSS), 2017

[https://en.wikipedia.org/wiki/Open\\_source](https://en.wikipedia.org/wiki/Open_source)

<https://computersweden.idg.se/2.2683/1.703485/microsoft-kop-github>

# OSS & samhällsutvecklingen

- Operativsystem
- Desktop-appar, Webb-appar
- Infrastruktur: språk, verktyg, CI, "Molnet"  
(serverhallar, datalagring)
- AI, ML
- Tvärindustriell utveckling  
(energi, transport, finans, media, ...)
- Den offentliga sektorn, **demokratin**  
public money ⇒ open source

# Hur välja OSS?

- Koden
  - api-kvalitet
  - dokumentation
  - mognad
  - aktivitet
- Gemenskapen *community health governance*
  - ledning
  - vem är aktiv med vad
  - code-of-conduct
- Affärsmodeellen
  - licensmodell (pryl, system, tjänst)
  - differentierande eller stödjande

# Hur få inflytande över OSS?

- meritokrati
- bidra aktivt, bygga förtroende
- skapa allianser
- samarbeta *och* konkurrera samtidigt

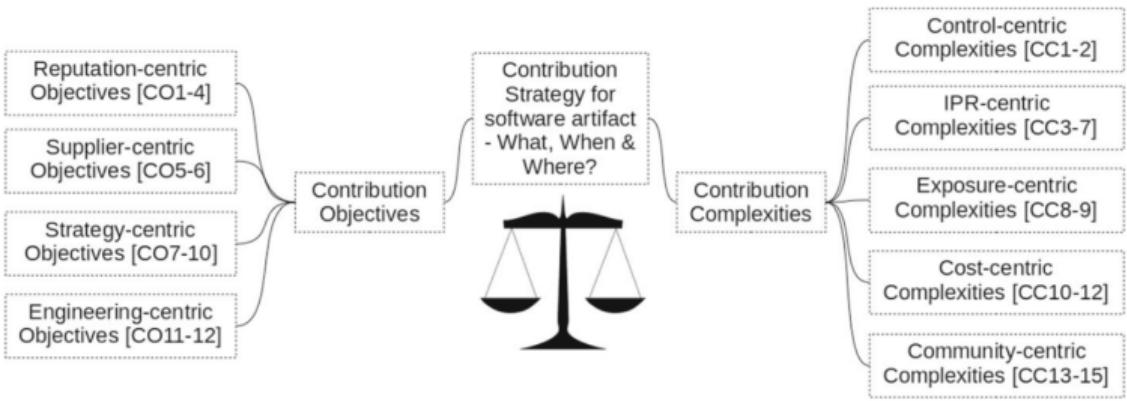
# Utveckla en företagsstrategi för att bidra till öppen källkod

**"What to share, when, and where:  
balancing the objectives and complexities of open  
source software contributions"**

*Linåker, J. & Regnell, B.*

*Empirical Software Engineering, (2020) 25:3799-3840*

<https://doi.org/10.1007/s10664-020-09855-2>



**Fig. 2** This study presents 27 considerations (12 Contribution Objectives (CO) and 15 Contribution Complexities (CC)) that may need to be considered by an organization when deciding on a contribution strategy for a software artifact. The COs and CCs are divided into four and five categories respectively and are listed in Tables 4 and 5

[cs.lth.se/krav](http://cs.lth.se/krav)