SOFTWARE ENGINEERING RETHOUGHT

Intelligence Amplification in Software Engineering

Three examples from research and practice

Michael Unterkalmsteiner (mun@bth.se)

SERL Sweden
LEADING SOFTWARE ENGINEERING

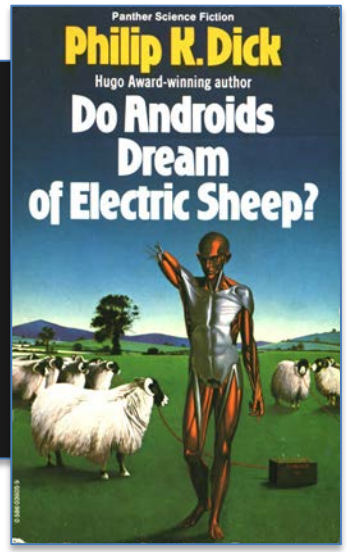
KK-stiftelsen

Michael Unterkalmsteiner

- Researcher - Engineer, PhD (Tekn. Dr.) Software Engineering, M.Sc. SE, B.Sc. Applied CS
- Senior lecturer at BTH (fulltime)
- Consultant at Qualicen GmbH (funtime ☺)
- Specialization on Requirements Engineering and Testing/Verification
  - Information flow / Communication in large companies
  - Assessments and problem identification
  - Solutions based on process changes and/or technical innovations
- Experience and published in
  - Process Assessment
  - Natural Language Processing
  - Data Mining and Machine Learning

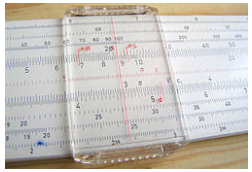
Intelligence Amplification in Software Engineering

Artificial Intelligence vs. Intelligence Amplification

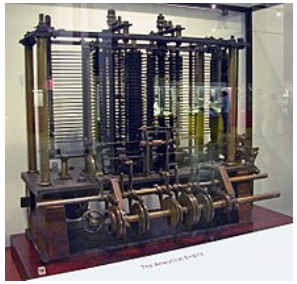


IA has been and is (nearly) everywhere

2300 BC



1620 AD



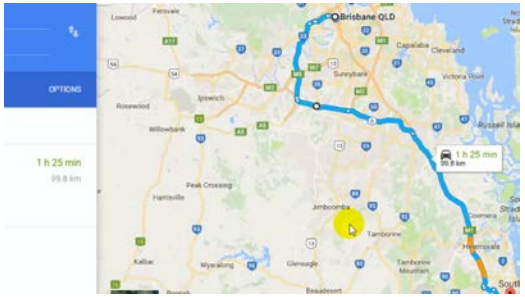
1837

1951

1989

1990s

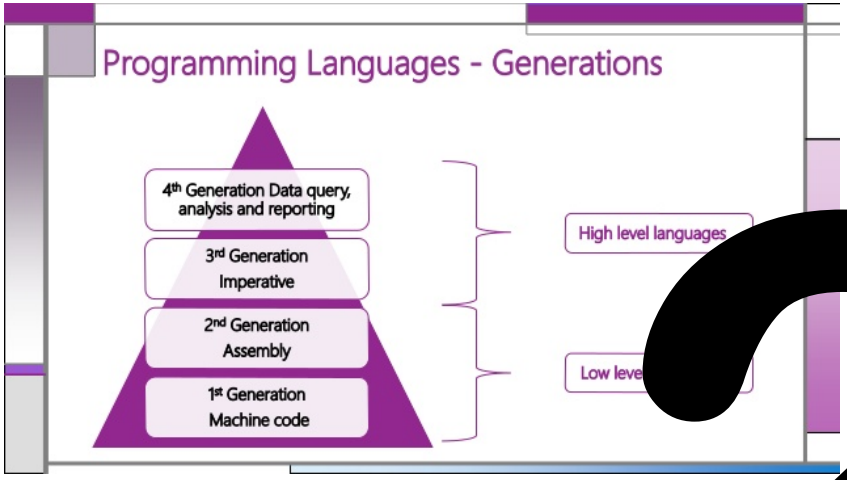
2001

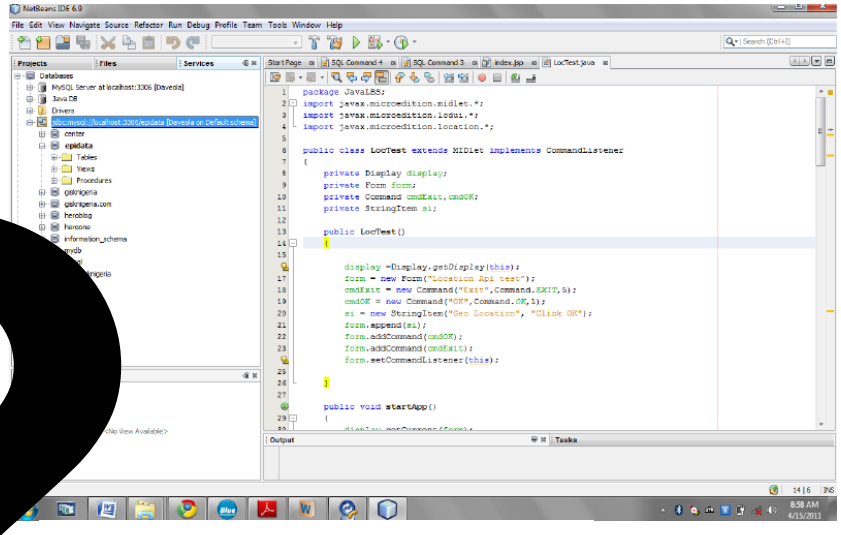


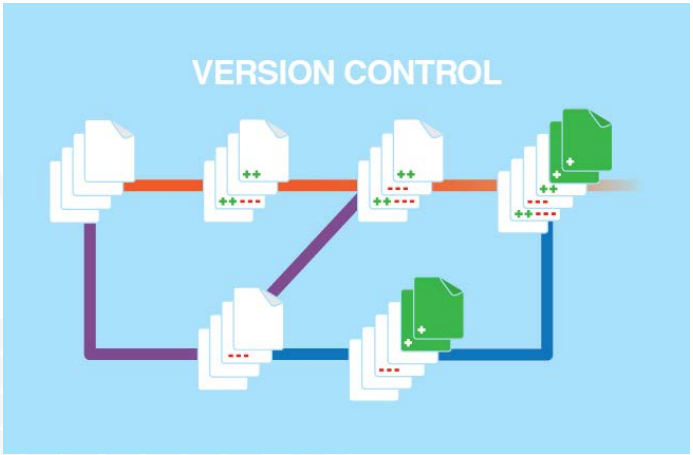
2004

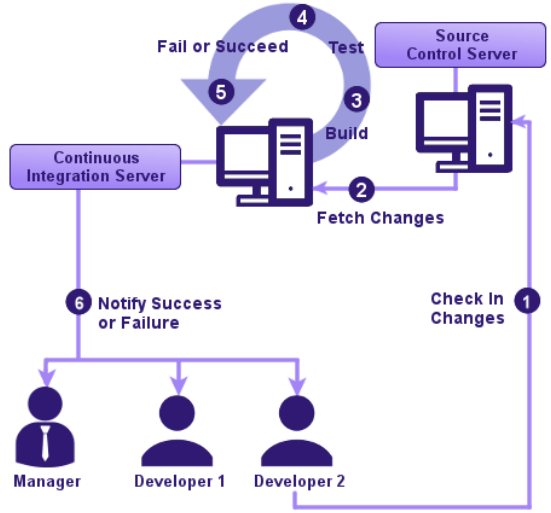
2010s

IA in Software Engineering









The missing links: examples of IA in SE

1. Do we test enough or too much?
2. Are our requirements good enough?
3. Do engineers understand the problem domain?
4. … [your questions here]

For each example we look at:

- WHY is this a problem/challenge
- HOW does the solution work
- WHAT is (could be) the future

Future IA in Software Engineering focuses on human-human interaction, with support from the computer.

Example 1: Specmate

Problem: How many test cases do I need to test this requirement?

- Effort estimation
- Time management and resource allocation

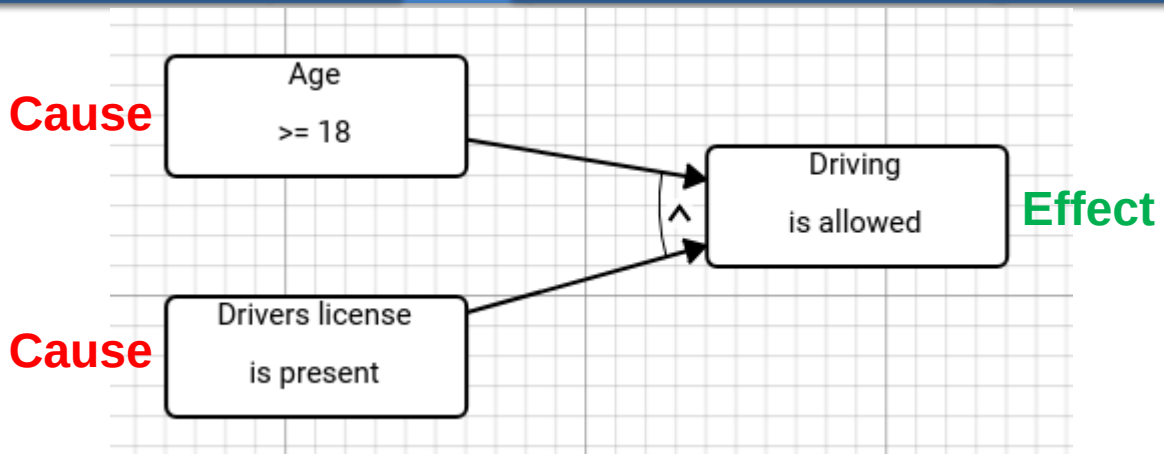
In collaboration with:
Maximilian Junker, Technical University Munich
Sebastian Eder, Technical University Munich

Example 1: Specmate

Problem: How many test cases do I need to test this requirement?

3

A person is allowed to drive a vehicle if the person is older than 18 years and possesses a driving license.



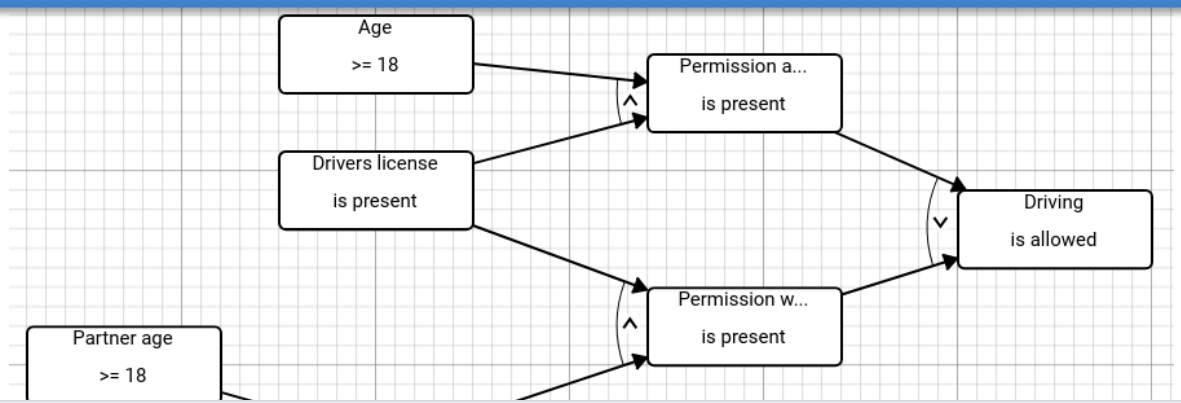
		Input			Output
Test Case	**Test Procedure**	Age	Drivers license	+	Driving
TestCase-1	+ Create test procedure	>= 18	not is present		not is allowed
TestCase-2	+ Create test procedure	not >= 18	is present		not is allowed
TestCase-3	+ Create test procedure	>= 18	is present		is allowed

Example 1: Specmate

Problem: How many test cases do I need to test this requirement?

5

A person is allowed to drive a vehicle if the person is older than 18 years and possesses a driving license. In addition, the person is allowed to drive even he/she is not yet 18 years old, if the person is accompanied by another person that is older than 18 years and that possesses a driving license.



		Input					Output
Test Case	**Test Procedure**	Age	Drivers license	Partner age	Partner drivers licens	+	Driving
TestCase-1	+ Create test procedure	not >= 18	is present	>= 18	is present		is allowed
TestCase-2	+ Create test procedure	>= 18	not is present	>= 18	is present		not is allowed
TestCase-3	+ Create test procedure	>= 18	is present	not >= 18	not is present		is allowed
TestCase-4	+ Create test procedure	not >= 18	is present	>= 18	not is present		not is allowed
TestCase-5	+ Create test procedure	not >= 18	is present	not >= 18	is present		not is allowed

Summary: Specmate

- **Problem:** How many test cases do I need to test this requirement?
  - Effort estimation
  - Time management and resource allocation
- **Solution:** model the requirements (light weight) and let the computer generate the optimal number of test cases
- **Positive side-effect:** if requirement cannot be modeled, it is likely that it is not properly understood → quick feedback to RE
- **Maturity level:** pre-production
- **Current investigation:** Do test professionals accept the modeling technique?
- **Future:** Improve efficiency. To what extent…
  - … can we generate models from textual requirements?
  - … can we reuse existing models?
- **Tool available at**: https://github.com/junkerm/specmate

Example 2: Requirements quality

In collaboration with:
Henning Femmer, Technical University Munich

Example 2: Requirements quality

- **Problem:** Natural language requirements are very common, but have issues: imprecise, ambiguous, compound, not testable, redundant, etc.
  - Thousands of requirements statements, often written by domain experts, but not requirements analysts
- **Traditional solution:** manual reviews using checklists and rules that are applied by requirements editors
- **Downsides:**
  - Much effort spent on trivial issues (proper referencing, consistent naming, ambiguous terminology, structure and tense of sentences)…
  - … leaving less time for hard issues: contradictions (internal and external), correctness, testability, etc.
- **Our approach:**
  - Studied ~150 requirements writing rules
  - Implemented 15 generic rules using Natural Language Processing technologies

Requirements writing rules examples

Kategori: Kravuppbyggnad

Regel: Subjektet ska komma tidigt i kravet

Makes it easier to identify the imporant aspect in the requirement.

Exempel (rätt / fel)

Utrymningsskyltar med belysning ska vara placerade ovan dörrar.

Vid större differens än 0,3 m ska **beslut om åtgärd, återplacering eller omkonnektering** ske i samråd med Trafikverkets förvaltare av spårgeometrier i GAD.

Kategori: Kravuppbyggnad

Regel: Första gången av förkortning†

The first time an abbreviation is used, it should also be defined.

Kategori: Referens

Regel: Dokumentnummer ska förtecknas under rubrik "Åberopade dokument"†

Reduces the probability of incorrect or not existing references.

Kategori: Nyckelord

Regel: Skrivningar liknande "det ska vara möjligt" ska undvikas

Reduces ambiguty and the probability of misunderstandings

Exempel (rätt / fel)

Systemet ska möjliggöra aktivering av sektioner av PA-system i form av högtalargrupper.

Det ska vara möjligt att aktivera sektioner av PA-system i form av högtalargrupper.

Kategori: Nyckelord

Regel: "/" ska användas i betydelse "eller", inte "och"

Reduces ambiguty and the probability of misunderstandings

Exempel

Att uppfylla frekvensband 900/1800/2100/2600 MHz, innebär att Entreprenör kan uppfylla 900 eller 1800 eller 2100 eller 2600 MHz, vilket inte var avsikten.

Kategori: Dokumentuppbyggnad

Regel: Alla figurer/tabeller ska förses med beskrivning†

Improves the understandability of the overall specification.

Real-time check of rule violations

Krav 4. Samtliga kemiska produkter, även bränslen (uppdelat på respektive drivmedelsslag), smörjmedel, hydrauloljor, som används i leveransen (förutom rengöringsmedel och förbrukningsvaror på kontor) ska finnas förtecknade och på begäran redovisas till Beställaren.

Krav 5. Förteckning över kemiska produkter ska omfatta:

a) Produkt- eller varunamn, kemiskt namn samt i förekommande fall CAS-nummer.

b) Klassning enligt Trafikverkets kemikaliegranskningsfunktion (A-C).

c) Förbrukad mängd eller volym för leveransen.

d) Aktuellt 16-punkters säkerhetsdatablad (på svenska) enligt REACH-förordningen.

Krav 6. Förteckningar med tillhörande säkerhetsdatablad ska finnas tillgängliga på arbetsplatsen.

Krav 7. Kemiska produkter ska förvaras och hanteras på sådant sätt att spill och läckage tas om hand och inte orsakar skada eller olägenhet för människors hälsa eller miljön.

Krav 8. Kemikaliespill ska saneras och hanteras som farligt avfall.

Krav 9. Kemiska produkter ska förvaras inlåsta, oåtkomligt för tredje man, när arbete inte sker på arbetsområdet.

Krav 10. Absorberande medel ska förvaras lättillgängligt där kemikalier hanteras.

5.1.3 MILJÖLEDNING

Krav 11. Leverantören ska ha ett miljöledningssystem som följer standard {SS-EN ISO 14001 Miljöledningssystem - Krav och vägledning} eller likvärdigt.

5.1.4 LEVERANTÖRENS MILJÖPLAN

Krav 12. Leverantörens miljöplan, vilken kan vara integrerad i projektplanen, ska uppfylla kraven i dokument {TDOK 2016:0032 Kvalitetsstyrning i upphandlad verksamhet – Entreprenad och Projekteringstjänster}.

5.1.5 LEVERANTÖRENS MILJÖREVISION

Krav 13. Miljöronder för kontroll av Beställarens krav och lagkrav ska genomföras minst var 14:e dag.

Krav 14. Beställarens representant ska kallas till miljöronder.

Smell: Subjektet ska komma tidigt i kravet. (Re
Exempel:
Krav 12: Utrymningsbelysning i utrymningsvägar ska alltid vara tänd.
Krav 13: Utrymningsskyltar med belysning ska vara placerade ovan dörrar.

Smell: Passiv voice (Regel 13)
Kravspecifikationens krav ska normalt skrivas så att de kravställer funktioner i den färdiga anläggningen eller hur den färdiga anläggningen ska vara utförd (tidsform Futurum exactum), dvs. kravet gäller inte bara vid övertagande utan även under garantitiden.

Smell: Passiv voice (Regel 13)
Kravspecifikationens krav ska normalt skrivas så att de kravställer funktioner i den färdiga anläggningen eller hur den färdiga anläggningen ska vara utförd (tidsform Futurum exactum), dvs. kravet gäller inte bara vid övertagande utan även under garantitiden.

Smell: Passiv voice (Regel 13)
Kravspecifikationens krav ska normalt skrivas så att de kravställer funktioner i den färdiga anläggningen eller hur den färdiga anläggningen ska vara utförd (tidsform Futurum exactum), dvs. kravet gäller inte bara vid övertagande utan även under garantitiden.

Smell: Undvik skrivningar innehållande "ska ha"
Fel: Krav 71. Tappvarmvattenberedare ska ha volym 40 l.
Rätt: Krav 71. Tappvarmvattenberedare ska vara dimensionerad för en volym om 40 l.
Strömningsskydd ska uppfylla skyddsklass vätskekategori 3. (ska ha)

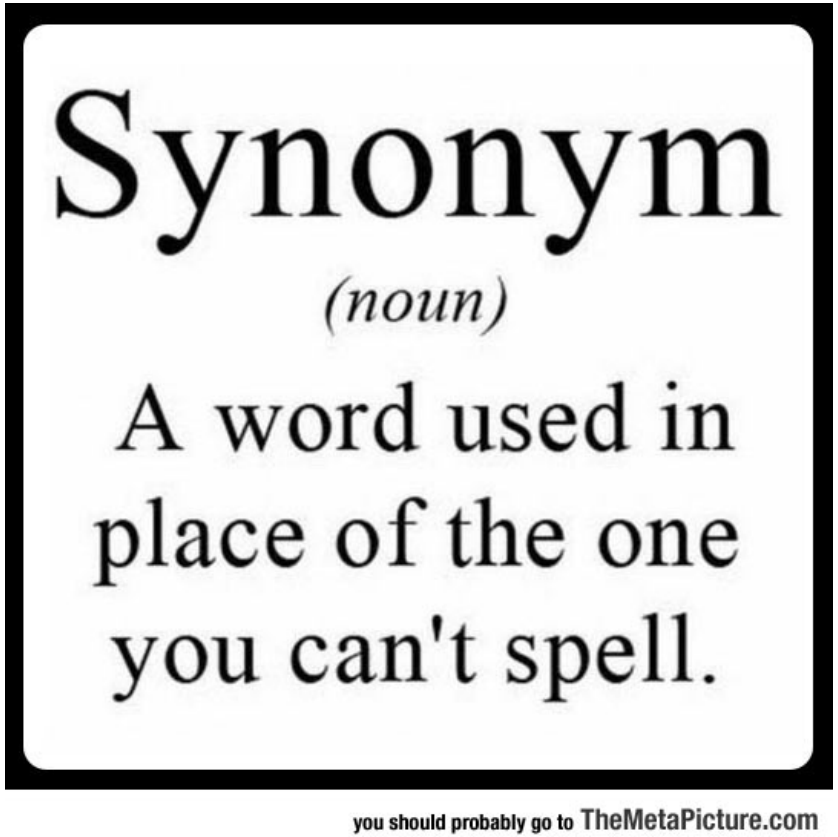
Smell: Referenser ska bli gulmarkeras (Regel 3
Under utarbetande av dokument ska alla referenser gulmarkeras då dokumentnummer och namn är under förändring. ({SS-EN ISO 14001 Miljöledningssystem - Krav och vägledning})

Smell: Krav avseende att något ska uppfylla en
Rätt: Krav 72. Pendlar ska vara enligt standard {SBF 120:7 § 17.2.3}.
Fel: Krav 72. Pendlar ska klara kraven i standard {SBF 120:7 § 17.2.3}.
Fel: Krav 72. Pendlar ska uppfylla standard {SBF 120:7 § 17.2.3}. (standard {SS-EN ISO 14001 Miljöledningssystem - Krav och vägledning})

Summary: Requirements quality

- **Problem:** Natural language requirements are very common, but have problems: imprecise, ambiguous, compound, not testable, imprecise, etc.
- **Our solution:** “spell-checker on steroids”: let the machine find the “easy” problems and let humans work on the “hard” issues.
- **Positive side effects:**
  - formally specify what quality means, hence it is conveyable and learnable
  - similar quality across projects; quality defined and managed centrally
- **Maturity level:** proof-of-concept
- **Current investigation:** How effective is the technique (false positive rate)?
- **Future:** Tap into quality aspects that are “hard” to automate: e.g. contradictive, compound, testable requirements

Example 3: Domain-specific synonyms



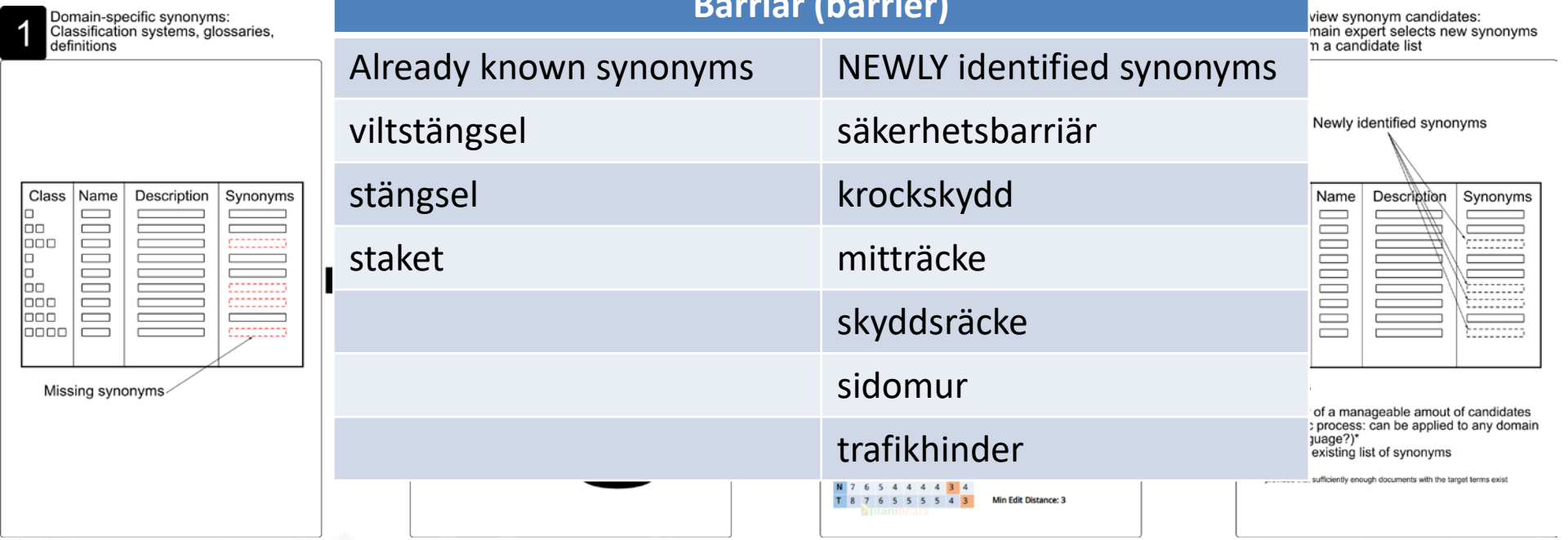
In collaboration with:
Andrew Yates, Max Planck Institute for Informatics

Example 3: Domain-specific synonyms

- **Problem:** different people call the same thing by different names: these are synonyms.
  - Domain-experts talking with software developers
  - Finding information (searching), e.g. in requirements, contracts, laws, policies
  - Inconsistencies in documentation -> NOT an issue if synonyms are known
- **Challenge:** find synonyms efficiently (work intensive if domain-specific, general thesauri are of no use)
- **Solution:** use (big)data-mining to identify a short-list of synonym candidates, let human expert decide which ones are actual synonyms

Example 3: Domain-specific synonyms

Domain: Construction and building business
CoClass classification system to identify and manage building components



Barriär (barrier)	
Already known synonyms	NEWLY identified synonyms
viltstängsel	säkerhetsbarriär
stängsel	krockskydd
staket	mitträcke
	skyddsräcke
	sidomur
	trafikhinder

Summary: Domain-specific synonyms

- **Problem:** different people call the same thing by different names: these are synonyms.
- **Solution:** exploit co-occurrence/similarity of terms to deduce meaning
  - Still challenging to construct a relevant corpus
  - Improve precision, i.e. the number of synonym candidates one has to look at
- **Maturity-level:** research experimentation and evaluation
- **Current investigation:** How many *new* synonyms does the technique identify?
- **Future:** Exploit learned associations between terms for other purposes

Summary of IA in Software Engineering

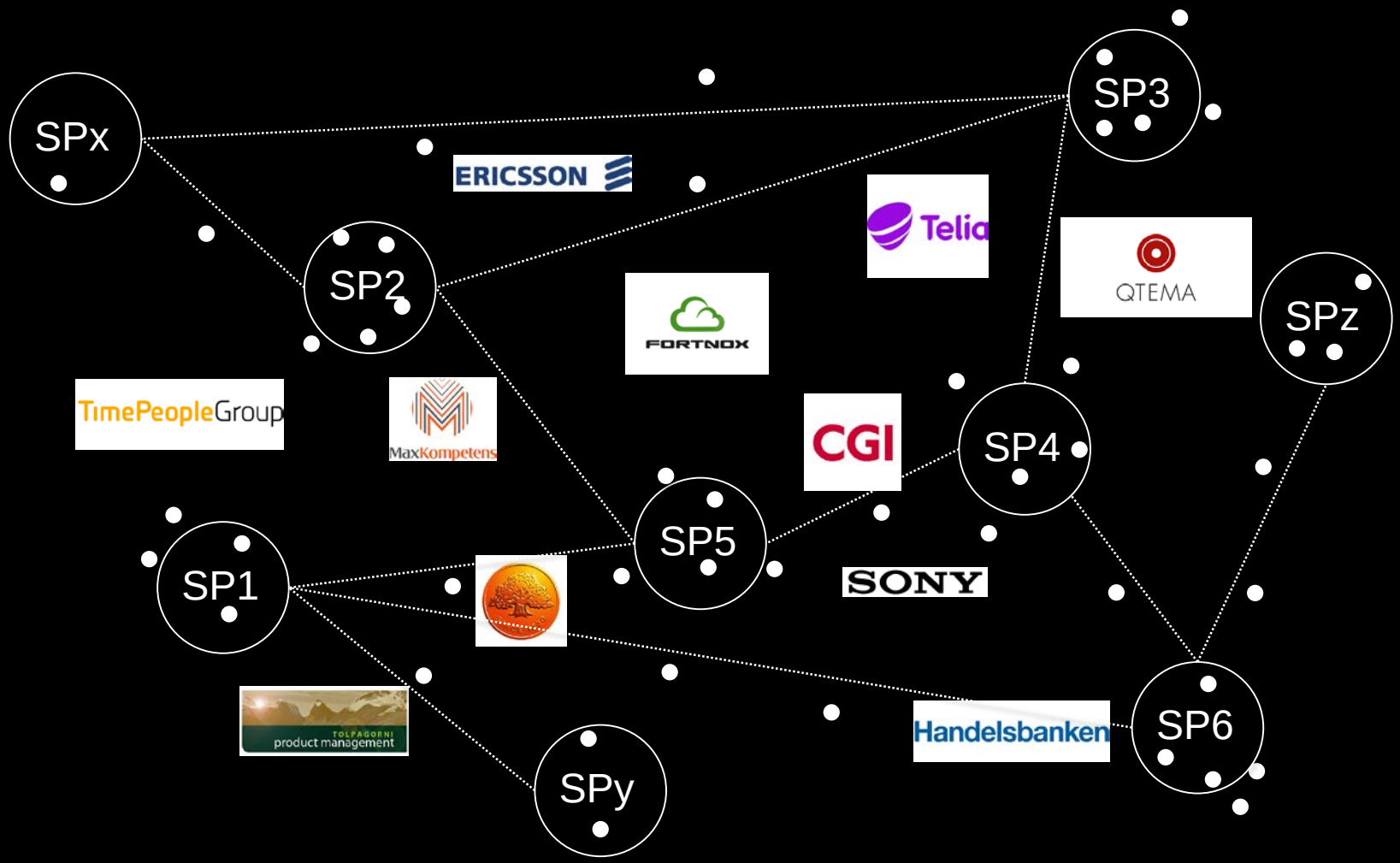
- Common traits of examples:
  - Human is always involved, machine does what it does best
  - Problems are narrow, so are the solutions
  - Tasks need to be known and defined very well (to a reasonable degree) in order to be able to provide IA
  - Benefit is effort reduction which translates into giving human more time to THINK.
    - Potential consequence: build a better product
- Differences of examples:
  - At varying research maturity levels
  - At varying technological maturity levels
  - Required support to use the technology

"The future is already here - it's just not evenly distributed"

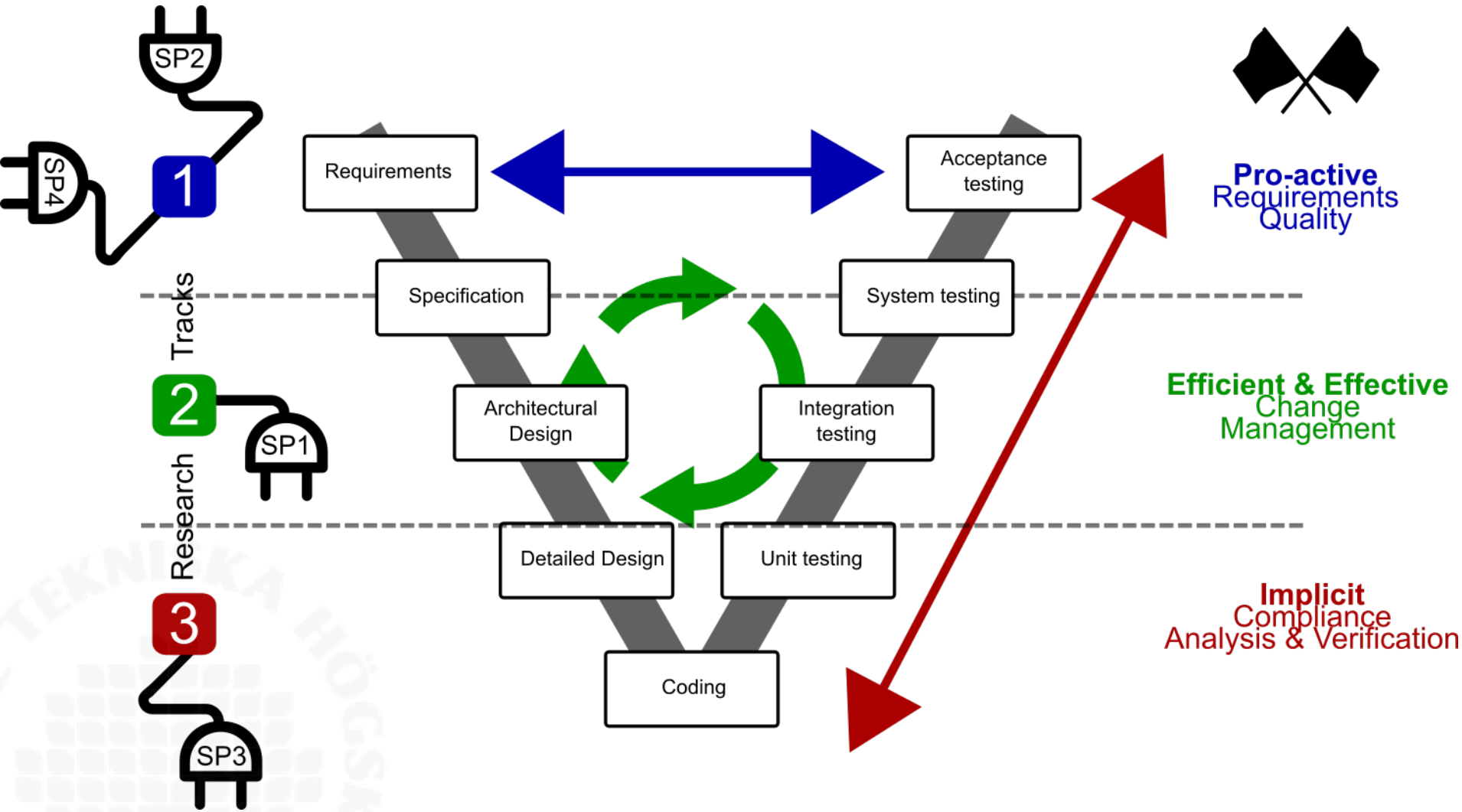
William Gibson

You were there when it started…

Software **E**ngineering **R**e**T**hought



Q&A

thank you
danke
謝謝
ngiyabonga
teşekkür ederim
gracias
merci
obrigado
bedankt
dziękuję
go raibh maith agat
sukriya
kop khun krap
terima kasih
감사합니다
ありがとう
спасибо
рахмат
Баярлалаа
tapadh leat
mochchakkeram
arigatô
takk
dakujem
grazie
taiku
sagolun
mauruuru
hvala
köszönöm
kiitos
dankie
dhanyavad
nandri
nanni
bayarlalaa
gràcie
enkosi
sobodi
dèkuji
mèsi
didi madloba
najis tuke
kam sah hamnida
তোমাকে ধন্যবাদ
rahmat
rahmet
tanemirt
diolch
dhanyavadagalu
shukriya
merce
мерси
trugarez
murakoze
chokrane
mamnun
дякую
djiere dieuf
tau
akun
dankon
aciu
spas
tack
welalin
barka
mersi
kia ora
vinaka
спасибі
blagodaram
faafetai lava
dank je
misaotra
matondo
paldies
grazzi
mahalo
хвала
asante
manana
obrigada
tenki
chnorakaloutioun
gratias ago
gràcies
sulpáy
xièxie
ευχαριστώ
ngiyabonga

SP6: Verification of Software Requirements in Dynamic, Complex and Regulated Markets