Wat is legacy software waard?



Wat is software?

Wat is de waarde van software?

Hoe haal je rendement uit software?

Over 15 minuten komt Gernot Eggen van ASML aan het woord over hetzelfde onderwerp.

geen tijd voor wetenschap; laat u meenemen door de retoriek.

ACICT 14 november 2024

Jurgen Vinju

CWI

TU/e

swat. engineering
control your software

Ethos

- **mens** sinds 1977 (Ermelo), opgegroeid in Zaanstad, toen Amsterdam, Parijs, New York, en Naarden. Vader.
- **professional** software engineer sinds Just Software v.o.f. anno 1997-2000



- **onderzoeker** Software Languages en Software Engineering anno 2000
- **onderwijzer** van Software Evolution UvA/VU/HvA, TUE, anno 2005
- **deeltijdhoogleraar** Automatische Software Analyse TU Eindhoven anno 2014



- **mede-eigenaar** Swat.engineering BV anno 2017
- **projectleider** allerlei open-source projecten anno 2005



- **winnaar** van academische prijzen (trouwens, wetenschap is teamwork)
- **partner** industriële samenwerkingen IBM, ING, ABN-AMRO, Philips, Canon, ASML, ...

Pathos



Pathos



Pathos

- **Vak** *Software Engineering* is woest interessant, uitdagend en nuttig:



Pathos

- **Vak** *Software Engineering* is woest interessant, uitdagend en nuttig:
  - Maakt buitengewone **flexibiliteit** en **creativiteit** mogelijk.



Pathos

- **Vak** *Software Engineering* is woest interessant, uitdagend en nuttig:
  - Maakt buitengewone **flexibiliteit** en **creativiteit** mogelijk.
  - **Vals onderscheid** binnen het vak: “hard versus soft skills”



Pathos



- **Vak** *Software Engineering* is woest interessant, uitdagend en nuttig:
  - Maakt buitengewone **flexibiliteit** en **creativiteit** mogelijk.
  - **Vals onderscheid** binnen het vak: “hard versus soft skills”
  - **Bestaande software** is vele malen interessanter dan toekomstige software

Pathos



- **Vak** *Software Engineering* is woest interessant, uitdagend en nuttig:
  - Maakt buitengewone **flexibiliteit** en **creativiteit** mogelijk.
  - **Vals onderscheid** binnen het vak: “hard versus soft skills”
  - **Bestaande software** is vele malen interessanter dan toekomstige software
- Trouwe **Adviesbureau ICT toetsing** volger en supporter; “een noodzakelijk goed”, *voorkom waardeverlies*.

Pathos



- **Vak** *Software Engineering* is woest interessant, uitdagend en nuttig:
  - Maakt buitengewone **flexibiliteit** en **creativiteit** mogelijk.
  - **Vals onderscheid** binnen het vak: “hard versus soft skills”
  - **Bestaande software** is vele malen interessanter dan toekomstige software
- Trouwe **Adviesbureau ICT toetsing** volger en supporter; *“een noodzakelijk goed”, voorkom waardeverlies.*
- **Weg met Calimero!** Nederland was en is *beter* in software en digitalisering

Pathos



- **Vak** *Software Engineering* is woest interessant, uitdagend en nuttig:
  - Maakt buitengewone **flexibiliteit** en **creativiteit** mogelijk.
  - **Vals onderscheid** binnen het vak: “hard versus soft skills”
  - **Bestaande software** is vele malen interessanter dan toekomstige software
- Trouwe **Adviesbureau ICT toetsing** volger en supporter; “een noodzakelijk goed”, *voorkom waardeverlies*.
- **Weg met Calimero!** Nederland was en is *beter* in software en digitalisering
- **Eigenaarschap en soevereiniteit;** onze veiligheid en onze sociaal-economische gerechtigheid.

Pathos



- **Vak** *Software Engineering* is woest interessant, uitdagend en nuttig:
  - Maakt buitengewone **flexibiliteit** en **creativiteit** mogelijk.
  - **Vals onderscheid** binnen het vak: “hard versus soft skills”
  - **Bestaande software** is vele malen interessanter dan toekomstige software
- Trouwe **Adviesbureau ICT toetsing** volger en supporter; “een noodzakelijk goed”, *voorkom waardeverlies*.
- **Weg met Calimero!** Nederland was en is *beter* in software en digitalisering
- **Eigenaarschap en soevereiniteit;** onze veiligheid en onze sociaal-economische gerechtigheid.
- **Systemen van de overheid:** *woest interessant*, technisch, juridisch, macro én micro economisch.

Logos: Software heeft last van onbegrip

Logos: Software heeft last van onbegrip

- Software: niet direct te zien, horen of voelen, maar overal aanwezig en merkbaar.

Logos: Software heeft last van onbegrip

- Software: niet direct te zien, horen of voelen, maar overal aanwezig en merkbaar.
- Maar wat *is* het nou precies? En wat moet je ermee?

Logos: Software heeft last van onbegrip

- Software: niet direct te zien, horen of voelen, maar overal aanwezig en merkbaar.
- Maar wat *is* het nou precies? En wat moet je ermee?
- Metaforen vliegen rond, maar slaan (bijna) altijd de plank mis.

Logos: Software heeft last van onbegrip

Een gebouw!

Een trein!

- Software: niet direct te zien, horen of voelen, maar overal aanwezig en merkbaar.

Engineering!

- Maar wat *is* het nou precies? En wat moet je ermee?

Kunst!

- Metaforen vliegen rond, maar slaa[illegible] mis.

Een bedrijf!

Architectuur!

Een brug!

Een vliegveld!

Een motor!

wetenschap!

Logos: Software heeft last van onbegrip

- Software: niet direct te zien, horen of voelen, maar overal aanwezig en merkbaar.
- Maar wat *is* het nou precies? En wat moet je ermee?
- Metaforen vliegen rond, maar slaan (bijna) altijd de plank mis.

Logos: Software heeft last van onbegrip

- Software: niet direct te zien, horen of voelen, maar overal aanwezig en merkbaar.
- Maar wat *is* het nou precies? En wat moet je ermee?
- Metaforen vliegen rond, maar slaan (bijna) altijd de plank mis.
- Hopen op **overdraagbaarheid** van best practises en resultaten via een metafoor = kansloos.

Logos: Software heeft last van onbegrip

- Software: niet direct te zien, horen of voelen, maar overal aanwezig en merkbaar.
- Maar wat *is* het nou precies? En wat moet je ermee?
- Metaforen vliegen rond, maar slaan (bijna) altijd de plank mis.
- Hopen op **overdraagbaarheid** van best practises en resultaten via een metafoor = kansloos.
- Onbegrip leidt tot **verlies aan grip**

Logos: Software heeft last van onbegrip

- Software: niet direct te zien, horen of voelen, maar overal aanwezig en merkbaar.
- Maar wat *is* het nou precies? En wat moet je ermee?
- Metaforen vliegen rond, maar slaan (bijna) altijd de plank mis.
- Hopen op **overdraagbaarheid** van best practises en resultaten via een metafoor = kansloos.
- Onbegrip leidt tot **verlies aan grip**
  - (Operationele en financiële) risico's niet kunnen inschatten

Logos: Software heeft last van onbegrip

- Software: niet direct te zien, horen of voelen, maar overal aanwezig en merkbaar.
- Maar wat *is* het nou precies? En wat moet je ermee?
- Metaforen vliegen rond, maar slaan (bijna) altijd de plank mis.
- Hopen op **overdraagbaarheid** van best practises en resultaten via een metafoor = kansloos.
- Onbegrip leidt tot **verlies aan grip**
  - (Operationele en financiële) risico's niet kunnen inschatten
  - Trends missen, bottlenecks komen als verrassing, plotselinge afschrijvingen, ...

Software is: kettingreactie van computerinstructies

Dit is geen metafoor, maar een beschrijving met illustraties.

de kunst is uit deze online versie in goed overleg verwijderd

Kunst van Mark Bischof

Film van Hugo Rompa

Software is: kettingreacties van computerinstructies

Software is: kettingreacties
van computerinstructies

Elke instructie:
leest, **rekent**, **schrijft**,
en **selecteert** de volgende.

Software is: kettingreacties
van computerinstructies

Elke instructie:
leest, **rekent**, **schrijft**,
en **selecteert** de volgende.

geautomatiseerde
oorzaken met gevolgen

Software is: kettingreacties
van computerinstructies

Elke instructie:
leest, **rekent**, **schrijft**,
en **selecteert** de volgende.

geautomatiseerde
oorzaken met gevolgen

de uitkomsten **ontstaan**
stap-voor-stap

Software is: kettingreacties
van computerinstructies

Elke instructie:
leest, **rekent**, **schrijft**,
en **selecteert** de volgende.

geautomatiseerde
oorzaken met gevolgen

de uitkomsten **ontstaan**
stap-voor-stap

Slechts beperkt door onze eigen
fantasie:
Software kan theoretisch alles
uitvoeren wat wij kunnen bedenken.

Software kettingreacties zijn vaak uniek voor hun *tijd* en *plaats* in de wereld

Fortnite

ASML's software

Ideal betalingen

Gemeentelijke basisadministraties

De Belastingdienst

Defensie

Wat hebben software systemen dan nog gemeen?



Wat hebben software systemen dan nog gemeen?



Software staat altijd model voor domeinkennis

Wat hebben software systemen dan nog gemeen?



Software staat altijd model voor domeinkennis

De Belastingdienst *n'est pas* de Belastingwet

Wat hebben software systemen dan nog gemeen?





Software staat altijd model voor domeinkennis

De Belastingdienst *n'est pas* de Belastingwet

Wat hebben software systemen dan nog gemeen?





Software staat altijd model voor domeinkennis

12 salarismaanden *n'est pas* 13 x 4 weken verloning

(Oude) wetten van software engineering

(Oude) wetten van software engineering

Lehman (1979)

“software verandert constant” en “software groeit altijd”

(Oude) wetten van software engineering

Lehman (1979)
“software verandert constant” en “software groeit altijd”

Conway (1984):
“software versmelt met, en weerspiegelt, bedrijfsorganisatie”

(Oude) wetten van software engineering

Lehman (1979)
“software verandert constant” en “software groeit altijd”

Conway (1984):
“software versmelt met, en weerspiegelt, bedrijfsorganisatie”

Vinju (2024):
“domeinkennis domineert bij software onderhoud”

Software moet veranderen



Software die niet verandert wordt snel waardeloos

Legacy software is al vaak veranderd

Software is groot en complex

Software is groot en complex

onoverzichtelijk, maar niet rommelig

Software is groot en complex

onoverzichtelijk, maar niet rommelig

arbitrair, maar niet per ongeluk

Software is groot en complex

onoverzichtelijk, maar niet rommelig

arbitrair, maar niet per ongeluk

repetitief maar toch uniek

Software is groot en complex

onoverzichtelijk, maar niet rommelig

arbitrair, maar niet per ongeluk

repetitief maar toch uniek

precair en vertoont “emergent” gedrag

Software is groot en complex

onoverzichtelijk, maar niet rommelig

arbitrair, maar niet per ongeluk

repetitief maar toch uniek

precair en vertoont “emergent” gedrag

opgedeeld in componenten die weer *geïntegreerd* worden

Veranderen is een werkwoord

Veranderen is een werkwoord

Ingrijpen in een unieke kettingreactie

Veranderen is een werkwoord

Ingrijpen in een unieke kettingreactie

Analyseren: “hoe en waarom” werkt dit nu?

Veranderen is een werkwoord

Ingrijpen in een unieke kettingreactie

Analyseren: “hoe en waarom” werkt dit nu?

Kennis en waarden van de organisatie reflecteren

Veranderen is een werkwoord

Ingrijpen in een unieke kettingreactie

Analyseren: “hoe en waarom” werkt dit nu?

Kennis en waarden van de organisatie reflecteren

Risico’s kunnen inschatten en ze inperken

Veranderen is een werkwoord

Ingrijpen in een unieke kettingreactie

Analyseren: “hoe en waarom” werkt dit nu?

Kennis en waarden van de organisatie reflecteren

Risico’s kunnen inschatten en ze inperken

Prognoses en rapportages kunnen maken

Veranderen is een werkwoord

Ingrijpen in een unieke kettingreactie

Analyseren: “hoe en waarom” werkt dit nu?

Kennis en waarden van de organisatie reflecteren

Risico’s kunnen inschatten en ze inperken

Prognoses en rapportages kunnen maken

Allemaal volkomen ondenkbaar zonder *expertise*

1. Software systemen moeten meeveranderen

1. Software systemen moeten meeveranderen

2. Software systemen zijn complex en uniek

1. Software systemen moeten meeveranderen

2. Software systemen zijn complex en uniek

3. Expertise van het domein en het systeem is “key enabler”

1. Software systemen moeten meeveranderen

2. Software systemen zijn complex en uniek

3. Expertise van het domein en het systeem is “key enabler”

dus...

1. Software systemen moeten meeveranderen

2. Software systemen zijn complex en uniek

3. Expertise van het domein en het systeem is “key enabler”

dus...

Legacy systemen zijn net zoveel waard als de expertise die er (nog) over bestaat.

Het CV van de software expert

Het CV van de software expert

in-house: voelt urgentie en waarden van de organisatie; staat aan de kant van het systeem.

Het CV van de software expert

in-house: voelt urgentie en waarden van de organisatie; staat aan de kant van het systeem.

analytisch: weet wat ze niet weet, kan antwoorden vinden in code en data, prognoses, risico's

Het CV van de software expert

in-house: voelt urgentie en waarden van de organisatie; staat aan de kant van het systeem.

analytisch: weet wat ze niet weet, kan antwoorden vinden in code en data, prognoses, risico's

"**lui en slim**"; grote effecten behalen in weinig tijd, zet innovatieve technieken in, durft hulp te vragen

Het CV van de software expert

in-house: voelt urgentie en waarden van de organisatie; staat aan de kant van het systeem.

analytisch: weet wat ze niet weet, kan antwoorden vinden in code en data, prognoses, risico's

"**lui en slim**"; grote effecten behalen in weinig tijd, zet innovatieve technieken in, durft hulp te vragen

klein ego; heeft geen heilige huisjes, kan feedback ontvangen en er haar voordeel mee doen.

Het CV van de software expert

in-house: voelt urgentie en waarden van de organisatie; staat aan de kant van het systeem.

analytisch: weet wat ze niet weet, kan antwoorden vinden in code en data, prognoses, risico's

"**lui en slim**"; grote effecten behalen in weinig tijd, zet innovatieve technieken in, durft hulp te vragen

klein ego; heeft geen heilige huisjes, kan feedback ontvangen en er haar voordeel mee doen.

eeuwige student en **docent**; kennisontwikkeling en -deling. Leert van externe en interne collega's

Legacy systemen zijn net zoveel waard als de expertise die er (nog) over bestaat.

Legacy systemen zijn net zoveel waard als de expertise die er (nog) over bestaat.

Expertise is makkelijker te managen dan een magisch “systeem”

Legacy systemen zijn net zoveel waard als de expertise die er (nog) over bestaat.

Expertise is makkelijker te managen dan een magisch “systeem”

Hoe herontdek je expertise in je organisatie en je systeem?

Legacy systemen zijn net zoveel waard als de expertise die er (nog) over bestaat.

Expertise is makkelijker te managen dan een magisch “systeem”

Hoe herontdek je expertise in je organisatie en je systeem?

Hoe verdeel je expertise over ruimte en tijd?

Legacy systemen zijn net zoveel waard als de expertise die er (nog) over bestaat.

Expertise is makkelijker te managen dan een magisch “systeem”

Hoe herontdek je expertise in je organisatie en je systeem?

Hoe verdeel je expertise over ruimte en tijd?

CWI, TUE, Swat.engineering: innovatieve taaltechnologieën; betere communicatie tussen code en experts

Uit het “maakbare systeem” frame, naar het “continue onderhouden” frame

Uit het “maakbare systeem” frame, naar het “continue onderhouden” frame

Zeg niet: ik heb een verouderd systeem!

Maar zeg: de expertise over ons systeem is niet meer bij ons.

Uit het “maakbare systeem” frame, naar het “continue onderhouden” frame

Zeg niet: ik heb een verouderd systeem!

Maar zeg: de expertise over ons systeem is niet meer bij ons.

Zeg niet: Carla doet het wel want die weet precies hoe het werkt!

Maar zeg: Carla mag het uitleggen aan Jack, want die moet het nog leren.

Uit het “maakbare systeem” frame, naar het “continue onderhouden” frame

Zeg niet: ik heb een verouderd systeem!

Maar zeg: de expertise over ons systeem is niet meer bij ons.

Zeg niet: Carla doet het wel want die weet precies hoe het werkt!

Maar zeg: Carla mag het uitleggen aan Jack, want die moet het nog leren.

Zeg niet: we moeten onze legacy laten vernieuwen

Maar zeg: we gaan onze expertise terugwinnen en borgen.

Uit het “maakbare systeem” frame, naar het “continue onderhouden” frame

Uit het “maakbare systeem” frame, naar het “continue onderhouden” frame

Vraag niet: wat is de nieuwste technologie die ze leren?

Maar vraag: welke onderhoud competenties leren ze?

Uit het “maakbare systeem” frame, naar het “continue onderhouden” frame

Vraag niet: wat is de nieuwste technologie die ze leren?

Maar vraag: welke onderhoud competenties leren ze?

Zeg niet: ons systeem is onbegrijpelijk… laat maar zitten

Maar zeg: we gaan ons systeem opschonen, voor meer kennis, lagere risico’s en kosten.

Uit het “maakbare systeem” frame, naar het “continue onderhouden” frame

Vraag niet: wat is de nieuwste technologie die ze leren?

Maar vraag: welke onderhoud competenties leren ze?

Zeg niet: ons systeem is onbegrijpelijk… laat maar zitten

Maar zeg: we gaan ons systeem opschonen, voor meer kennis, lagere risico’s en kosten.

Zeg niet: ons systeem moet toch innovatief zijn?

Maar zeg: we gaan innovatieve onderhoudstechnieken inzetten
(bijv. code=data, DSLs, MBT, MDE, LLMs)

Uit het "maakbare systeem" frame, naar het "continue onderhouden" frame

Vraag niet: wat is de nieuwste technologie die ze leren?

Maar vraag: welke onderhoud competenties leren ze?

Zeg niet: ons systeem is onbegrijpelijk… laat maar zitten

Maar zeg: we gaan ons systeem opschonen, voor meer kennis, lagere risico's en kosten.

Zeg niet: ons systeem moet toch innovatief zijn?

Maar zeg: we gaan innovatieve onderhoudstechnieken inzetten
(bijv. code=data, DSLs, MBT, MDE, LLMs)

Zeg niet: aanbesteding? out-sourcing? want wij zijn klein en zij zijn groot…?

Maar zeg wél: eigen expertise! samenwerking! co-creatie! ACICT, CIO Rijk, ICTU, …,
NWO-I CWI, MBO, HBO, Universiteiten, *MKB, horizontale dynamiek*, en *soevereiniteit*

https://www.markbischof.com/nl/ Google Lens Relaunch to update

MARK BISCHOF

Deze onbeschrijfelijk mooie kettingreacties werken alleen als ze onderhouden worden door de eigenaar en kunstenaar

Vragen zijn welkom!



software=kettingreacties

software=normatief

software verandert

software groeit

software versmelt

domein-expertise is de sleutel

niemand schrijft ons de software voor

Atelierbezoek in Amsterdam mogelijk tot 15 pax

mark@markbischof.com