Computeren som værktøj i forskning: er vi gode håndværkere?

Tobias Lindstrøm Jensen, Specialkonsulent, tlj@its.auu.dk og
Ellen Vibeke Knudsen, fuldmægtig, evk@aub.auu.dk

Ansatte ved universitetet bruger mange timer foran en computer, f.eks. med udvikling af kvantitative analyser og simuleringer af modeller. Når man sidder der bag skærmen, og forvandler viden til brød, kan det måske være opbyggeligt og fordelagtigt at overveje sin arbejdsform. Ikke blot det videnskabelige men også "er jeg effektiv/ineffektiv?", "jeg observer en effekt, men er den grundet i den beskrevne metode og data eller en fejl i koden?", "føler jeg mig på sikker videnskabelig grund?".

Først i det senere år, er der for alvor kommet en øget opmærksomhed omkring brugen af computeren i forskning. At man i sit eget felt er fagligt velfunderet, er en selvfølgelighed, men hvad sker der når man bliver smidt foran en computer? Denne problematik kan måske bedst blive beskrevet via et eksempel. Der er kommet en ny ph.d.-studerende eller post-doc ind ad døren, og man har sagt "du kan godt bruge en computer. Du skal udvikle det her stykke software med 10000 linjers kode. Just do it".

Det er lidt som at sige til en ny tømmerlærling "du kan bruge en hammer, så du skal bygge dette hus." Selv hvis det er muligt at bygge huset indenfor deadline og den givne økonomiske ramme, så kan der opstå problemer omkring kvalitet, reglementer, vedligeholdelse osv. Mange af de samme tanker gør sig også gældende når der skal bygges software.

Derfor er kompetenceudvikling indenfor disse områder yderst vigtig. Især når vi gerne tænker vores arbejde ind i større videnskabelige tankegods som forsvarlighed, velunderbygget, reproducerbarhed osv. Man kan spørge sig selv og kollegaerne: er vores arbejde med vores videnskabelig software på et niveau der modsvarer vores høje tanker om forskning?

Software Management Planer

Mange har ifm. deres forsknings eller biblioteksarbejde de senere år stødt på begreber og metoder tilknyttet Data Management Planer (DMP). DMPer er dokumenter, der beskriver ens datahåndteringspraksis for at belyse de problemstillinger, der er tilknyttet til datahåndtering. Derfor er det også blevet et vigtigt redskab og påbudt element til fondsansøgninger til f.eks. EU. I kølvandet på DMP opstod der lignende tankegang omkring software hvor Software Management Planer (SMP) kan være et værktøj til at belyse nogle af de problemstillinger der eksisterer ved videnskabelig softwareudvikling. Problemstillinger kan være placering af kode, sikkerhed, udviklingsmetoder, ansvarsfordeling, dokumentation, versionsstyring og versionering, licenser, og holdbarhed - altså kan vi stadig reproduce resultater og udvikle vores software når nu den ph.-d.-studerende eller post-doc har fået et nyt arbejde?

SMP har ikke haft den samme opbakning og politiks medvind som DMP de senere år, men flere rådgiver om at tage disse i brug af egen fri vilje, for f.eks. at forbedre impact, kvalitet eller effektivitet, også fra fondene. Dette er også forstærket af, at der er kommet flere gode templates når en SMP skal færdiggøres. Enkelte fondsansøgningskald lister SMP'er som obligatorisk, men mere vigtigt er det dog at det f.eks. er forslået til National Science Foundation (NSF) samt af The National Aeronautics and Space Administration

(NASA) at SMP'er skal med i fondsansøgninger på linje med DMP'er. Hvis disse store institutioner vælger SMPer som et vigtigt og måske obligatorisk element, vil disse tanker sikkert brede sig. Hvis man vil stå stærkt i fremtidigt

forskningsarbejde og ansøgninger, kan det være en god ide at få styr på praksis indenfor den videnskabelige softwareudvikling.

Internationale tiltag

I de senere år er der kommet en øget fokus på alle aspekter af videnskabelig softwareudvikling, og disse tanker og strømninger har resulteret i flere nye tiltag rundt om i verdenen. Den største organisation der forsøger at understøtte disse forandringer er The Carpentries med et mål om at opbygge grundlæggende kompetencer indenfor data og beregningsfærdigheder globalt. Men også regionale tiltag arbejder med de samme grundtanker, eksempelvis Software Sustainable Institute i Storbritannien, Better Scientific Software og US Research Software Sustainability i USA og CodeRefinery i de nordiske lande.

Motivationen bag disse organisationer er at forbedre produktivitet og kvalitet indenfor e-science ved at udvikle en bedre forskningspraksis, bl.a. ved at udskifte dårlige eller uproduktive metoder og værktøjer indenfor den anvendte praksis. Mange forskere har gennem mange år opnået et højt niveau indenfor egen faglighed, bl.a. ved at modtage undervisning, læsning, sparring, netværk, feedback osv., men er til en vis grad autodidakt indenfor udvikling af videnskabeligt software. Autodidakte softwareudviklere kan give problemer, også indenfor forskning. Disse problemer bliver adresseret med materiale og undervisning hvor deltagerne lærer nogle nye metoder, netværker med ligesindede, indgår i erfaringsudveksling, og dermed få en bedre platform at stå på. Både ift. det videnskabelige arbejde og produktet men også videreudvikling af egen kompetencer indenfor udvikling af videnskabeligt software. Forhåbentlige kan nye ideer og viden omkring metoder og værktøjer også sprede sig i vandet ved de enkle forskningsgrupper.

En undervisningspraksis der ofte anvendes i disse organisationer, er individuelt deltagende live kodning. Ved denne læringsform forstås at underviseren viser alle skridt for at udføre en bestemt opgave, og deltagerne udfører lignende skridt for at genskabe resultaterne på egen computer. Det kunne være udvikling af kode til et bestemt formål, udvikling af test, opsætning af værktøjer osv. Ideen er at synliggøre arbejdsprocessen, se hvordan erfarne "håndværkere" håndterer problemer og fejl samt opbygge et vist niveau af muskelhukommelse. Igen ligesom en lærling følger en svend og ser processen, hvordan problemer håndteres og indlærer hvordan opgaverne kan udføres.

En anden tilgang der ofte anvendes i undervisning, og bliver tilskyndet er fagfællebedømmelse (peer-review). Det kan lyde mærkeligt, men fagfællebedømmelse af understøttende software er meget mindre udbredt i forskningskredse end bedømmelse af forskningsmanuskripter. Men mange vil nok nikke genkendende til, at kvaliteten af det software man udvikler, nok vil være højere, hvis man beslutter at være helt åben omkring det udviklede produkt og lade andre vurdere dette. Desuden kan feedback, både fra gode kollegaer men også unavngivne kritikere, være brugbar ifm. ens egen udvikling og kompetenceopbygning.

Brug et godt værktøj

Når man anvender det rette værktøj til en opgave, har det både betydning for effektivitet samt kvalitet. Et værktøj og metode, der med fordel kan anvendes ifm. videnskabelig softwareudvikling, er versionsstyring. Mange kender problemet og irritationen ved at have filer med navne som "presentation_rev01.pdf", "presentation_final.pdf", "presentation_final2.pdf". Disse filnavne er et udtryk for en manuel og usystematisk versionsstyring.

Med en usystematisk versionsstyring er det svært at adresser og svare på spørgsmål som "hvad er den nyeste version og hvornår er den præcis fra?", "hvilke ændringer er indført mellem to versioner?", "hvem lavede disse ændringer?" "en tidligere version af dette program virkede, men det gør det ikke længere. Hvad gør jeg?", "hvorfor lavede

jeg en ændring sidste måned?", "hvordan arbejder jeg samtidig med andre på samme kode?". En mere systematisk tilgang til versionsstyring med brug af dedikeret værktøjer, giver gode muligheder for at svare på alle disse spørgsmål. Disse problemstillinger findes ikke blot til arbejde omkring softwareudvikling, men kan videreføres til arbejde på artikler, ansøgninger, præsentationer, undervisningsmateriale osv.

En interessant vinkel ved gode versionstyringsværktøjer er, at man kan lege arkæolog. Forfatterne har f.eks. prøvet at grave gammel kode frem for at kunne genskabe en metode og reproduce en figur i en anden artikel. Dette kan kun lade sig gøre, hvis der er styr på versionerne. Som arkæolog er man nu temmelig sikker på, at man har et stykke kode ved hånden som er lig med det som de andre forfattere havde til rådighed, da artiklen blev skrevet. Med metoden ved hånden, blev det også klart, at den ikke kunne ikke løse alle de problemer, der var beskrevet i artiklen. Der var et misforhold, som først blev åbenlyst og beviseligt da man kunne køre koden. Dermed er versionstyringsarkæologi en vigtig metode til at skabe klarhed over, hvordan eksisterende resultater er frembragt.

En anden meget anvendelig metode med tilhørende støttende værktøjer er test. Test af videnskabeligt software kan bedst sammenlignes med kalibrering. Hvis maskinerne ikke er kalibreret, er målingerne ubrugelige. Ligeledes burde et ikke testet stykke software være ubrugelig. Test er også vigtig ift. tillid. Når nogle resultater dukker op skærmen, vil man sikkert spørge sig selv – "har jeg tillid til resultaterne?". Her står test som et vigtigt redskab til netop at opbygge tillid til resultaterne. Det næste spørgsmål er sikkert "Hvordan sikre jeg mig, at vejledere og fagfæller også har tillid til resultaterne?" Her kan test dokumentation være et vigtigt værktøj.

Ud over kompetenceudvikling indenfor versionsstyring og test, bliver der også ofte arbejdet med metoder og værktøjer til data håndtering og organisering, deling af kode og data, samarbejde, dokumentation, automatiske testmiljøer, scripting, licenser og reproducerbarhed.

Referencer:

Kald:
EPSRC, "High End Computing (HEC) Consortia Call", (2017)
EPSRC, "Computational Science and Engineering: Software for the Future II", (2014)

Forslag:
NASA, "Open Source Software Policy Options for NASA Earth and Space Sciences", (2018)
NSF, "CI2030: Future Advanced Cyberinfrastructure: A report of the NSD Advisory Committee for Cyberinfrastructure" (2018)
ERC Scientific Council, "Open Research Data and Data Management Plans: Information for ERC grantees" (2019)

Templates:
The Software Sustainability Institute, "Checklist for a Software Management Plan", DOI: 10.5281/zenodo.1422656.
"PRESOFT (Preservation for REsearch SOFTware) project", http://www.france-grilles.fr/presoft-en/

Baggrund
The Software Sustainability Institute, "Software Management Plan", https://www.software.ac.uk/software-management-plans
Simon Hettrick, "Research Software Sustainability: Report on a Knowledge Exchange Workshop", Knowledge Exchange, (2016)

R Bast: A FAIRer future, https://www.nature.com/articles/s41567-019-0624-3
Merali, Z: Computational science:… error https://www.nature.com/news/2010/101013/pdf/467775a.pdf

D. L. Donoho (2010). An invitation to reproducible computational research, Biostatistics, 11(3), pp. 385-388, https://doi.org/10.1093/biostatistics/kxq028

G.K. Sandve et al. "Ten Simple Rules for Reproducible Computational Research".In: PLoS Computational Biology 9.10 (2013), https://doi.org/10.1371/journal.pcbi.1003285