DIKU 11. juni 2012
Erik Frøkjær

Eksamensopgave i kurset
Projektkursus Systemudvikling 2012

Aflevering: Sted og tidspunkt
Besvarelsen af eksamensopgaven, kaldet eksamensrapporten, skal afleveres fredag den 22. juni 2012 kl. 14-16 på **DIKU, Universitetsparken 1, 2100 København Ø**, i **sofagruppen, stueetagen i sydfløjen**. Her vil instruktor Andreas Frisch (og/eller en anden af kursets instruktorer) være tilstede [Andreas Frischs telefonnummer, for det tilfælde at du ikke kan finde stedet, er: 3020 6192]

Afleveringen omfatter *både* (a) papirkopi *og* (b) elektronisk upload:

(a) Antal papirkopier
To papirkopier (et til eksaminator og et til censor). Hvis du ønsker kvittering for afleveringen, bedes du medbringe en fotokopi af opgavens forside.

(b) Elektronisk upload
Eksamensrapporten inklusiv bilag uploades i Absalon under punktet "Eksamensopgaven 2012". Eksamensrapporten uploades som *een samlet pdf-fil* senest fredag den 22. juni 2012 kl 16.00.

Størrelse
Eksamensrapporten *uden* bilag må maksimalt være på 15 sider (A4 med normalt layout, ca. 2400 tegn/side). Figurer og bilag medregnes ikke i det maksimale sidetal.

Sprog
Du kan vælge at skrive på dansk (norsk eller svensk) eller engelsk.

Forside
Forsiden *skal* indeholde følgende oplysninger:
- Titel på IT-projektet
- Kursusbetegnelsen: "Projektkursus Systemudvikling 2012"
- Dit navn og de første 6 cifre af dit cpr-nummer
- Navne på de personer, som har været med i din projektgruppe
- Link til Youtube-præsentation (se nedenfor: Opgavetekstens punkt 1.6)

Hvis det med jeres brugere er aftalt, at projektet skal behandles fortroligt, skal eksamensrapporten på forsiden påtrykkes teksten "F O R T R O L I G".

Generelt om besvarelsen
Eksamensrapporten skal være en *individuel* besvarelse. Som det fremgår nedenfor baserer eksamensrapporten sig i nogen udstrækning på resultater, som er dokumenteret i projektgruppens fire obligatoriske delrapporter. Hver enkelt gruppemedlem kan ved udarbejdelsen af sin eksamensrapport frit benytte sig af materialer fra disse delrapporter - uden ændringer eller i revideret form.

Bedømmelse af eksamensrapporten: Eksamensrapporten vil blive bedømt på, hvorledes du er i stand til at give en klar, sammenhængende og dækkende besvarelse af ovenstående spørgsmål. Som led heri indgår, om du effektivt kan benytte begreber og resultater fra kursuslitteraturen (se oversigt i

bilag A). Der bliver således lagt vægt på den skriftlige fremstilling, dvs. om eksamensrapporten er klar, velorganiseret og læsevenlig. Vær derfor omhyggelig med at skrive, illustrere, redigere og korrekturlæse eksamensrapporten.

Indhold
Eksamensrapporten skal give en besvarelse af nedenstående tre spørgsmål med tilhørende underspørgsmål. Du skal på egen hånd fortolke opgaveteksten, som den foreligger. I eksamensperioden vil der ikke fremkomme supplerende eller korrigerende oplysninger til opgaveteksten, ligesom undervisere og instruktorer *ikke* er tilgængelige for besvarelse af spørgsmål.

Der ønskes en besvarelse af alle spørgsmål hver for sig. Ved besvarelsen af det enkelte spørgsmål kan der henvises til diverse bilagsmaterialer og til besvarelsen af andre spørgsmål. Henvisninger skal angives præcist med bilagsnummer, sidenummer, figurnummer osv.

Bilagene fungerer som mulig reference for læseren. Besvarelsen må altså *ikke forudsætte* læsning af bilag. For at kunne holde eksamensrapporten indenfor det nævnte sidetal er det vigtigt, at besvarelsen koncentrerer sig om det væsentlige. Undgå fx at benytte 1-2 sider på en detaljeret beskrivelse af en kompliceret funktionalitet, hvor i stedet nogle få linjer og en henvisning til bilag kunne gøre det.

Eksamensrapporten skal have en indholdsfortegnelse, der også omfatter bilag. Alle eksamensrapportens sider skal være fortløbende nummereret, også omfattende bilag.

Spørgsmål 1: IT-projektet (50%)
Dette spørgsmål omhandler det IT-system, du har været med til at udvikle i løbet af kurset. Beskriv projektet på basis af jeres besvarelse af Fjerde Delrapport. Giv en klar og sammenhængende beskrivelse af jeres IT-projekt i følgende punkter:

1.1 IT-projektets formål og rammer

1.2 Kravspecifikation for IT-løsningen
Under dette punkt skal kunne findes omtale af:
(a) De funktionelle og ikke-funktionelle krav til jeres system. Husk at angive eventuelle constraints.
(b) Et use case diagram, der beskriver system-funktionaliteten. Der ønskes et højniveau-diagram, som giver overblik over hvilke use cases, de forskellige aktører har.
(c) Tre specificerede use cases, som er særlig vigtige i jeres system.
(d) Et klassediagram over jeres problemområde (solution-domain).
(e) Sekvens-diagrammer over de 3 use-cases specificeret i punkt (c).
Husk at alle diagrammer *skal* være fulgt af tekstbeskrivelser, der gør diagrammerne fuldt forståelige også for læsere uden særligt domænekendskab.

1.3 Systemdesign sammenfatning

1.4 Program- og systemtest

1.5 Brugergrænseflade og interaktionsdesign
Under dette punkt skal kunne findes:

(a) Præsentation af skærmbilleder af de mest interessante dele af jeres brugergrænseflade.
(b) Illustration af flowet/dynamikken i brugerinteraktionen mellem skærmbillederne, gerne i form af et såkaldt 'navigationsdiagram', jf Fjerde Delrapport.
(c) Resultatet af seneste tænke-højt forsøg gennemført med én eller flere af jeres brugere.

1.6 Link til Youtube-præsentation

En audio-visuel præsentation af brugergrænsefladen af den seneste kørende prototype. (Formatet vælger I frit, fx en video der illustrerer IT-løsningen med vigtige use cases, en serie screenshots med speak, slideshow med forklarende tekst. Dog skal det i alle tilfælde afleveres som et YouTube link og være uploadet som en "skjult video". *Maksimal varighed: 5 minutter*).

1.7 Udvalgt implementationsarbejde

Udvælg en delopgave ved implementationsarbejdet, som du fandt særlig udfordrende eller interessant. Beskriv problemet, og hvordan du og projektgruppen valgte at løse det programmeringsmæssigt.

Spørgsmål 2: Om samarbejdet med brugerne og samarbejdet i projektgruppen (30%)

Dette spørgsmål omhandler samarbejdet mellem din projektgruppe og de eksterne parter i IT-projektet, primært brugerne; samt projektgruppens interne samarbejde og planlægning af arbejdet.

2.1 Giv en skematisk oversigt over din gruppes projektforløb fra indledende brugerkontakt, analyse, design, prototyper og evaluering heraf. Fremhæv i oversigten hvornår brugerkontakter/ møder foregik.

2.2 Beskriv forholdet mellem parterne i jeres projekt, fx udviklere, kunde, ledelse, fremtidige brugere, brugerrepræsentanter, andre berørte parter. Beskriv udvalgte forhold som fik stor betydning for projektforløbet, fx vanskeligheder med at forudsige kompleksiteten i brugernes krav og ønsker, kvalitetskrav til løsningen, forhandlinger og aftaler om projektets mål og omfang, og interessemodsætninger.

2.3 Beskriv samarbejdet internt i projektgruppen, fx vanskeligheder med planlægningen af arbejdet, at holde aftaler, og gensidig støtte, når nøgleopgaver ikke blev løst tilfredsstillende. Hvad blev gjort for at afhjælpe vanskelighederne?

2.4 Beskriv styrker og svagheder ved tilrettelæggelsen af jeres projektforløb og samspillet mellem systemudviklingsaktiviteterne - såsom analyse, design, implementation, prototyping, evaluering/ test, dokumentation, kommunikation og samarbejde. Diskuter dine erfaringer i forhold til *kursuslitteraturen*.

2.5 Giv begrundede forslag til forbedringer af jeres proces. Diskutér - og begrund ved brug af *kursuslitteraturen* - under hvilke betingelser dine forslag vil kunne overføres som retningslinjer for andre systemudviklingsprojekter.

Spørgsmål 3: Diskussion af en udvalgt artikel i forhold til kursuslitteraturen (20%)

Til slut i denne opgavetekst finder du en artikel med titlen 'No Silver Bullet: Software Engineering Reloaded', som resumerer hovedpunkter fra en paneldiskussion afholdt ved OOPSLA 2007 konferencen. Artiklen er skrevet af Steven Fraser og Dennis Mancl og blev publiceret i Software, IEEE, January/February 2008, p 91-94.

3.1 Giv et overblik over hovedsynspunkterne fra paneldiskussionen.

3.2 Undersøg og diskuter hvorvidt disse hovedsynspunkter er samstemmende med hvad du har kunnet læse i OOSE-bogen, i anden faglitteratur på kurset (jf Bilag A: Kursuslitteratur) eller erfaret gennem jeres IT-projekt.

Bilag A: Kursuslitteratur

Betegnelsen kursuslitteratur omfatter følgende:

Lærebogen (OOSE-bogen):

Bruegge & Dutoit (2010), *Object-Oriented Software Engineering - using UML, Patterns and Java*, Third edition, Pearson 2010. Chapter 1-16

Artikler oploadet på kursushjemmesiden i mappen 'Undervisningsmateriale/Artikler mv til Pensum':

Brooks, F. P. (1986), No silver bullet, *Proc. of the IFIP Tenth World Computing Conference*, edited by H.-J. Kugler (1986), pp. 1069-76. Reprinted version from the book Brooks, F. P. (1995), *The mythical man-month: Essays on software engineering*, Addison-Wesley Publishing.

Ehn, P. and Kyng, M. (1991), Cardboard computers: Mocking-it-up or hands-on the future. In Greenbaum, J. and Kyng, M., editors, (1991), *Design at Work: Cooperative Design of Computer Systems*, pages 169–195. Lawrence Erlbaum Associates.

Frøkjær, E. and Hornbæk, K. (2005), Cooperative usability testing: complementing usability tests with user-supported interpretation sessions. In *CHI '05 extended abstracts on Human factors in computing systems* (CHI EA '05). ACM, New York, NY, USA, 1383-1386.

Gould, J. D. and Lewis, C. (1985), Designing for usability: key principles and what designers think. *Commun. ACM* 28, 3 (March 1985), 300-311.

'OOAD Kap 1': Mathiassen, L., Munk-Madsen, A., Nielsen, P. A., Stage, J. (2001), *Objektorienteret Analyse & Design*, Forlaget Marko, 3. udg., 2001. Kapitel 1.
Molich, R. (2003), *User testing, Discount user testing*, DialogDesign, www.dialogdesign.dk

Naur, P. (1985), Programming as theory building, *Microprocessing and Microprogramming* 15, 253-261. Reprinted version from Naur, P. (1992), *Computing: a Human Activity*, ACM, pp. 37-49.

Parnas, D. L. and Clements, P. C. (1986), A rational design process: How and why to fake it. *IEEE Trans. Softw. Eng.* 12, 2 (Feb. 1986), 251-257. The original version from *Proceedings of the TAPSOFT conference*, Formal Methods and Software Development, Berlin 1985, pp. 80-100.

Features Editor: Dale Strok ■ dstrok@computer.org

No Silver Bullet: Software Engineering Reloaded

Steven Fraser and Dennis Mancl

A celebratory panel took place at the 22nd International Conference on Object-Oriented Programming, Systems, Languages, and Applications in Montreal. The occasion was the 20th anniversary of Fred Brooks' paper "No Silver Bullet: Essence and Accidents of Software Engineering." The paper appeared in the April 1987 *Computer*, reprinted from the *Proceedings of the IFIP 10th World Computer Congress* (North-Holland, 1986). The panelist positions appear in the OOPSLA 2007 *Conference Companion* (ACM Press).

Steve Fraser as impresario introduced the panel, which included Fred Brooks (Univ. of North Carolina at Chapel Hill), David Parnas (Univ. of Limerick), Linda Northrop (Software Engineering Inst.), Aki Namioka (Cisco Systems), Dave Thomas (Bedarra Research), Ricardo Lopez (Qualcomm), and Martin Fowler (ThoughtWorks).

Steve started by polling the audience: "How many of you have read the paper?" About three-quarters raised their hands. "I remember that it came out on the day of my doctoral defense. My thesis supervisor said it was a good thing that I didn't say anything that disagreed with Fred."

Opening statements

The first panelist to speak was Fred Brooks, who is widely recognized for his 1975 book *The Mythical Man-Month* (Addison-Wesley), a collection of essays on software project management. The book was based on Fred's experience as the project manager for the development of IBM's System/360 family of computers and then the OS/360 operating system and compilers. Fred recapped "No Silver Bullet," suggesting that software challenges are either essential or accidental. The premise of the paper was that unless the remaining accidental complexity is 90 percent of all the remaining complexity, shrinking all accidental complexity to zero still would not result in an order-of-magnitude improvement. Fred suggested that useful solutions must address inherent complexity—observing that object-oriented techniques have come closest to achieving this goal.

Next up was David Parnas, whose collected papers were published in *Software Fundamentals* (Addison-Wesley) in 2001. In particular, he's known for his *Communications of the ACM* papers "On the Criteria to Be Used in Decomposing Systems into Modules" (May 1972) and "Software Aspects of Strategic Defense Systems" (Dec.

0740-7459/08/$25.00 © 2008 IEEE

1985). David described a "silver bullet" as a technique that requires no training or experience to apply—a silver bullet should find its mark without aim. He stated that no such single technique exists. He asked two questions: "Why was it necessary for Fred to write 'No Silver Bullet?' Why talk about it when 20 years have proved it right? Designing software is hard and will always be hard. No easy answers here!" David suggested that a proportion of our community would sell wooden legs to a snake and then try to measure increased snake productivity. In order to sell whatever idea they believe in (and many honestly believe in the goodness of their product), they paint it as a silver bullet. Most programmers don't produce quality software, and they're always looking for better tools and quick fixes. David argued that progress made to date as a result of software tools and methods is a myth—suggesting that progress has been the result of hardware improvements rather than software improvements.

Linda Northrop commented that she was a university professor when "No Silver Bullet" was first published, and had promptly made it required reading in her classes. The article inspired students. We've made some progress in software development, as evidenced by the construction of more complex systems. Productivity has improved significantly in niche areas such as software product lines. Object orientation was about modeling, as Kristen Nygaard put it. However, Nygaard said we lost the battle when we missed the boat about essence in OO. So it's easier to work around the edges and address accidents. The essential difficulties are conformity, complexity, and changeability. We still need to cultivate discipline, great designers, and hard work.

Aki Namioka agreed with the premise of "No Silver Bullet" but remained optimistic. Software engineering as a discipline has expanded significantly, both in terms of system complexity and the community's global extent. Aki referenced the Second Life environment, the topic of an OOPSLA keynote talk. Second Life is an example of how programming is now accessible to many nondevelopers. Now it takes a "village" to ship software with many disparate stakeholders.

Dave Thomas reflected on OOPSLA's beginnings, his advocacy of OO, and how he had been impressed by Simula's elegance and by what could be done in Smalltalk. He commented that today's object technology is gratuitously complex (for example, middleware and frameworks), so much so that talented people spend their time trying to stay on top of things, because the technology is unstable and constantly changing. For many current applications, the basic problem hasn't changed. These applications can often be expressed as simple functional transformations.

Dave criticized the software industry for its adoption of certification programs, suggesting that industry has opted for certificates rather than competence. People don't have a sense of fundamentals and basic concepts. Objects and agility are great, but objects aren't everything. Product engineering is what's important, and domain-specific languages can be very useful to improve productivity. There's great potential for so-called "high-barrier" languages often associated with the functional- and vector-programming communities. These languages for power programmers have enabled real successes in niche domains and can be nicely extended via domain-specific languages. Google's MapReduce is an example of productivity gains derived from functional programming that hides the distributed infrastructure's complexity.

Ricardo Lopez believes that the fear of software failure drives a fruitless quest for silver bullets. We try to avoid the things we fear. Ricardo hypothesized that complexity is the way we've personified our fear, but to fear complexity is to fear life. Simplicity isn't something that's conserved (in nature); life is elegant in its complexity. The attempt to avoid complexity leads to accidents. Striving for excellence is the real silver bullet that will deliver an order-of-magnitude improvement through growth, both personal and professional. The silver bullet must come from within, rather than without. WE are THE Silver Bullet—which we achieve by professional excellence.

Martin Fowler stated that "No Silver Bullet" continues to be both an enjoyable must-read and an influential reference. In a blink (captured in a YouTube video), Martin was transformed into the Werewolf of Brooks' article.

As the Werewolf, he said, "I am alive and well"—and continued stating that OO was "an evil idea" but that he was able to overcome it. "Certainly there are many good ideas in object orientation, but the great thing is: nobody actually does it."

People use languages without the ideas, so the object community still has much to do. The Werewolf also has other weapons. "I love multicore concurrency systems." He felt that the use of prebuilt components was dangerous to him, but it was a theory with a crucial weakness: "It only helps you if the libraries are any good. I've been very good at getting people to build very bad libraries."

As for good designers, the Werewolf said that fortunately nobody outside the conference realizes that they're important. Software's invisibility helps him, and our challenge is to convince others that good designers are necessary. He expressed a liking for waterfalls and the tendency to generate unrealistic plans. The desire for silver bullets creates far more work for people.

Now it takes a "village" to ship software with many disparate stakeholders.

Dialogue with the audience

After the introductory remarks, Steve took audience questions. The first came from Bertrand Meyer (ETH), who commented that when "No Silver Bullet" first appeared, some managers used it as an excuse to avoid new technologies. Managers doubted many of the productivity claims made in the late 1980s for OO languages, design techniques, and tools. They could point to "No Silver Bullet" to back up their prejudices. Bertrand praised the brave people who advance the field of software engineering and who continue to promote their silver bullets.

■ Fred reiterated that technologies that

attack accidental difficulties won't solve the real problem.
- Linda thought that we need to focus on user needs, not just promote new technologies. She agreed that there are noble reasons to trumpet a silver bullet but that there's one ignoble one—greed!

John Roberts (Qualcomm) asked the panel if teamwork and collaboration might be a silver bullet.

- Ricardo and Dave Thomas both endorsed the idea of getting people to learn from each other. Agile development practices are one approach that might get your most experienced people to work with your young, inexperienced staff.
- David Parnas thought that anything that requires some learning time is really a "lead bullet." A silver bullet is something that any novice can use effectively with minimal training.
- The Werewolf pointed out with glee that "good people can't collaborate in teams." Most people don't work hard enough at getting teams to collaborate effectively, and heavyweight processes usually diminish collaboration.

Joe Yoder (The Refactory) advocated a combined approach for attacking the Werewolf: good people, understanding requirements changes, refactoring, good design, and teamwork. He called this approach "silver buckshot."

- Dave Thomas liked this approach but thought it wasn't repeatable enough to be a silver bullet. He added that leadership was also an important component of building a good team.
- Fred offered two books for guidance on leadership and growing a good team: *Peopleware* (2nd ed., Dorset House, 1999), the software engineering classic by Tom DeMarco and Tim Lister, and *The Carolina Way* (Penguin, 2004), by retired University of North Carolina basketball coach Dean Smith.
- Linda agreed that leadership was needed. She lamented that "we don't lead; we manage."
- The Werewolf was worried about *Peopleware*, which he called a "dangerous book." Fortunately, he mused, using MS Project is more alluring to managers. "Each time someone makes another Gantt chart or PERT chart, I get to eat another kitten."

The panel was asked for ideas to quantify the impact of new technologies:

- The Werewolf smirked. "There is no way to run repeatable experiments in software engineering to figure out if one method is better than another; you can just argue at conferences. This isn't really science."
- Ricardo agreed that productivity measurements are hard. No one likes lines of code as a measure. Agile methods sometimes measure output by counting the number of completed user stories, but some kind of weighting should be applied.
- David Parnas pointed out a measurement dilemma. Do we prefer a developer who produces 2,000 lines in two days or 500 lines in three days for the same task?
- Aki questioned the rationale behind measurement: "What are the goals that you want to achieve?" Instead of an assessment of productivity, managers might be using measurement as a pressure tactic to get developers to do more.

The panel was asked to clarify the difference between essential and accidental complexity. What can we do to get over the accidental issues and really get to the essence?

- Aki suggested that we've come a long way (mostly positive) with programming environments and debugging tools.
- The Werewolf saw a major communication obstacle. Developers are isolated because managers don't understand developers and because developers aren't good at talking with customers.

We've come a long way with programming environments and debugging tools.

Brian Foote (Industrial Logic) asked everyone to think about some "worse is better" notions. He claimed that somehow the world is working out, even though there is a lot of appalling code. A lot of the bad code actually works well enough to satisfy user needs. So, maybe the silver bullet is to make people better at writing bad code.

- Ricardo distinguished between bad code and cataclysmic code. He thought that systems should be designed to tolerate software failures, and we need a way to weed out "bad" code.
- David Parnas wasn't sure. "What's good for developers isn't necessarily good for the world. I know folks who have written software that only they can fix."

One questioner compared today's software technology advances with the history of mathematics. Calculus was a kind of silver bullet in the world of mathematics and physics, but going from geometry to calculus took over 1,500 years. If we think of OO as an unfinished silver bullet, how do we "finish" objects so that "real" people can use them?

- David Parnas was ready: "That is the topic of my talk this afternoon. You won't like the answer." His afternoon talk was titled "Precise Software Documentation: Making Object Orientation Work Better."
- Fred suggested, "Try to make a system good for one application, and then generalize."
- Ricardo warned about losing good technologies. He recalled that Archimedes developed calculus 2,000 years ago but that the Romans destroyed his work.
- Dave Thomas didn't endorse all var-

iations of object technology. Certain kinds of frameworks, "frameworks where insides leak out creating massive dependencies," aren't good. Packaging software as components can help. Many techniques can be used in the alchemy of building systems.

Last words

Because the panel's allotted time was running out, Steve had each panelist give a short summary:

- The Werewolf reminded the audience that the unexpected appearance of complexity is what gives him power. Humans always have so much optimism, and they overestimate their capabilities.
- Ricardo disagreed with the Werewolf. We're making progress against complexity every day. Of course, we're also creating new complexity. The complexity in our life 10 years from now will be different from today's complexity.
- Dave Thomas thanked Fred for the grand challenge. Even if we haven't achieved a silver bullet, the journey has been valuable and remains a challenge for the next generation.
- Aki suggested that there are both complex and simple systems to build, and not all situations require silver bullets.
- Linda echoed Fred with a philosophical observation: In life, we should focus on the essence, not the accidents.
- David Parnas defended the waterfall approach as originally proposed with inherent iteration. He recommended reading the original waterfall paper by Winston Royce, "Managing the Development of Large Software Systems" (*Proc. IEEE WESCON*, IEEE Press, 1970). Royce pointed out that the waterfall could never work without iteration and feedback. Parnas said the waterfall has been made a straw man to try to sell something better.
- Fred was worried that we don't understand enough about others' successes and failures. His point: "I know of no other field where people do less study of other people's work."

As panel impresario, Steve had the final word—thanking both the panelists and the audience for a spirited discussion and proclaiming, "There is no silver bullet!"

Steven Fraser is a Director (Engineering) at Cisco Research. Contact him at sdfraser@acm.org.

Dennis Mancl is a Distinguished Member of the Technical Staff at Alcatel-Lucent. Contact him at mancl@alcatel-lucent.com.

HOW TO REACH US

Writers

For detailed information on submitting articles, write for our Editorial Guidelines (software@computer.org) or access www.computer.org/software/author.htm.

Letters to the Editor

Send letters to

Editor, *IEEE Software*
10662 Los Vaqueros Circle
Los Alamitos, CA 90720
software@computer.org

Please provide an email address or daytime phone number with your letter.

On the Web

Access www.computer.org/software for information about *IEEE Software*.

Subscribe

Visit www.computer.org/subscribe.

Subscription Change of Address

Send change-of-address requests for magazine subscriptions to address.change@ieee.org. Be sure to specify *IEEE Software*.

Membership Change of Address

Send change-of-address requests for IEEE and Computer Society membership to member.services@ieee.org.

Missing or Damaged Copies

If you are missing an issue or you received a damaged copy, contact help@computer.org.

Reprints of Articles

For price information or to order reprints, send email to software@computer.org or fax +1 714 821 4010.

Reprint Permission

To obtain permission to reprint an article, contact the Intellectual Property Rights Office at copyrights@ieee.org.