TDDI02

Programmeringsprojekt. Föreläsning 1

Jonas Lindgren, Institutionen för Datavetenskap, LiU

På denna föreläsning:

- Kursinformation
- Vad är Software Engineering?
- Hur går ett projekt till?
- Anatomin hos en kravspecifikation

Kursledning

- Kursledare: Jonas Lindgren

- Handledare: Mattias Rönn
- Handledare: Rebecka Geijer Michaeli
- Handledare: Anton Sundblad

- Examinator: Jonas Wallgren
- Kursadministratör: Åsa Kärrman

Kursupplägg

Kursens mål:

- Programutveckling i *metodisk projektform*, inkl. dokumentation
- Kunskap om *några element* inom Software Engineering
- Kodning av lite *större volym*

Förkunskaper:

- God förtrogenhet med något ”högnivåspråk”
- Praktisk kunskap inom datastrukturer och algoritmer

Genomförande:

- 3 initiala föreläsningar
- 1 lektion, OOA/OOD
- Hemtenta
- 1 gästföreläsning av Kristian Sandahl
- 1 praktikföreläsning, användbara verktyg
- Projekt

Examination

- Projekt
  - Genomförs i grupper av 4
  - Kravspecifikation
  - Designspecifikation
    - Presentation
    - Granskning
  - Leverans
    - Demonstration
    - Kodinlämning
  - Testrapport (eventuellt..)
  - Erfarenhetsrapport
- Hemtenta
  - Individuell
  - 2 tillfällen till kompletteringar
- Kursen betygsätts med G

Övrigt

- Kursen kräver stort egenansvar!
- Konsultera hemsidan!
- Konsultera litteratur!
- Regelbundna möten med handledare är ett krav, ca. ett i veckan
  - Förmedla status via Kernel Alphas
  - Rapportera via e-mail: Vad som gjorts – av vem – tidsuppskattning
- Att göra en intern planering i form av en *projektplan* rekommenderas starkt!
- Börja bilda 4-personers grupper snarast!
  - 4 personer, varken mer eller mindre
  - För in er i webreg innan nästa fö., länk på kurshemsidan
- Deadline torsdag:
  - Gruppbildning klar
  - Val av projekt färdigt
  - Första handledarmötet genomfört

Projektförslag

- Krav - Minst ett externt bibliotek, ex:
  SDL2, SFML, Qt, MySQL, SQLite, ENet, Box2D, etc.

- Qt / MySQL / SQLite:
  - Textbaserat äventyrsspel med editor
  - Bibliotekshantering
  - Ligahantering

- SDL2 / SFML / Enet / Box2D:
  - ”Tower Defense”
  - ”Roguelike”
  - Risk
  - Racingspel

- Egna förslag:
  - Iphone- / Android-app?

Utgångspunkt

Programmering (= kodning) och design (= konstruktion) är teknikområden.

Framställning av stora/komplexa system kräver dessutom t.ex.

- Många programmerare/teams
- Personalfrågor (Specialister, utbildning, ersättare, etc..)
- Externa frågor (Marknadsföring, kontrakt, etc..)
- Kvalitetssäkring (Processer, produkter, etc..)
- Dokumentation av olika slag
- Management (Ledning, uppföljning, resursfördelning, etc..)

Till stor del icke-tekniska frågeställningar!

Utgångspunkt (cont.)

Området Software Engineering omfattar både dessa tekniska och icke-tekniska kompetenser.

Ett metodiskt arbetssätt i projektarbetsform behövs.

Ett projekt löper allmänt i ordningen:

1. Förstå problemet
2. Planlägg lösningen
3. Genomför planen
4. Utvärdera resultatet

Software Engineering?

En praktisk och vetenskaplig del av datalogi

- Som anger metoder, verktyg, riktlinjer, attityder, etc..
- För konstruktion av stora/komplexa programvarusystem
- I enlighet med användares/beställares intentioner
- Inom föreskrivna budget- och tids-ramar
- Med hänsyn till kvalitets- och underhålls-aspekter

”SE is the body of THEORY and PRACTICAL TECHNIQUES that can be brought to bear on the process of developing software.”

Software Engineering? (cont.)

Mest övergripande mål:

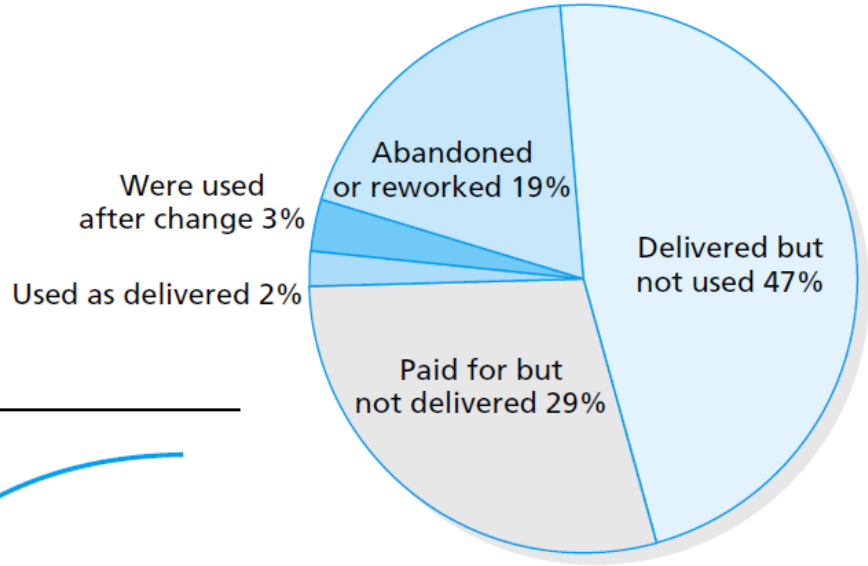
- Förmåga att urskilja och rätta sig efter kunders önskemål och krav (dvs. behovsstyrt, inte teknikstyrt!)
- Användande av ingenjörsmässiga principer, idéer, kvaliteter och attityder

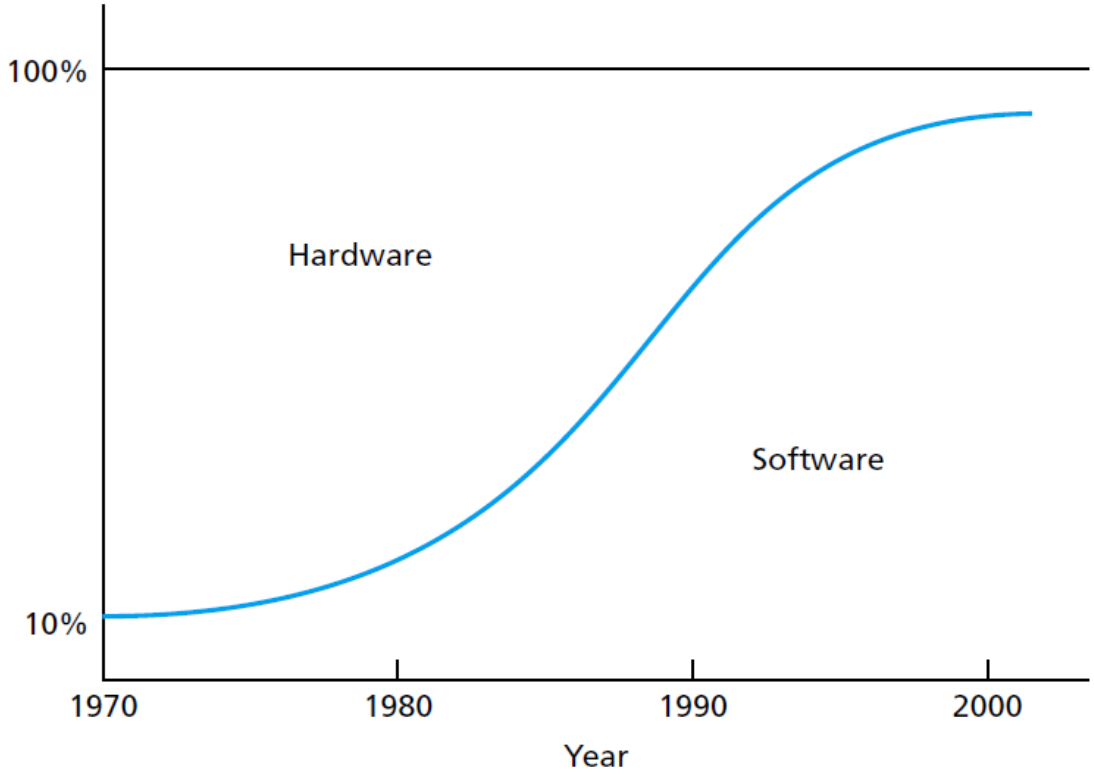
Vad innebär *ingenjörsmässighet*?

”Konstruktion av programvara är inte (längre) ett ’ad hoc’-jobb utfört av enstaka, kreativa individer i den mörka skrubben, utan ett välorganiserat, metodbaserat teamwork, baserat på känd teknik.”

Software Engineering? (cont.)

Källa: Software Engineering for Students: A Programming Approach, D. Bell





Traditionell arbetsgång

	Projektfas, allmänt	**SE-fas**	**Resultat**
1.	Förstå problemet	Kravanalys	Kravspecifikation
2.	Planlägg lösningen	Planering (tid, resurser)	Projektplan
3.	Genomför planen	Design (konstruktion), implementera	Designspec, ev. på flera nivåer, kod
4.	Utvärdera resultatet	Testning (validering, verifikation), flera nivåer	Ny kod, uppdaterade dokument

Dessa steg bryts ner i flera, mer preciserade, delsteg.

Projekt?

”En tillfällig kraftsamling (endeavour) som genomförs för att skapa en unik produkt, tjänst eller resultat.”

- Ett *definierbart* ändamål (ett *Opportunity*)
  - Definieras i en kravspecifikation: funktionalitet, prestanda, uppträdande, etc..
- Ett *unikt* företag
  - Inte ”rutinarbete”, avser inte något som gjorts identiskt tidigare
- En *tillfällig* aktivitet
  - Det finns en tydligt början och ett tydligt slut, enligt krav.

Kort definition:

”Ett projekt är en kombination av resurser som förs ihop för att skapa något som inte fanns förut.”, Cleland och Ireland, 2002

Begreppsdistinktioner

Principer (eller Practices):

- Ett enskilt förfaringssätt, sätt att arbeta
  - Parprogrammering
  - ”Planning Poker”

Metod:

- Ett konkret, detaljerat, förfaringssätt, inklusive verktyg och principer.
  - Extreme Programming (XP)
  - Jackson Structured Programming (JSP)

Metodik:

- ”Processmodell”er, med generella och gemensamma drag för ett flertal metoder.
  - Prototyping
  - Unified Process (UP, OpenUP)

Metodologi:

- ”Läran om” hur metoder konstrueras, kan värderas, dess generella egenskaper.

Begreppsdistinktioner (cont.)

Validering:

Med *validering* menar man att slå fast att det som levereras (eller interna dokument på vägen) har en *funktionalitet* i enlighet med det beställaren avsett. Det relaterar till kunden(en *Stakeholder*) och dennes förväntningar.

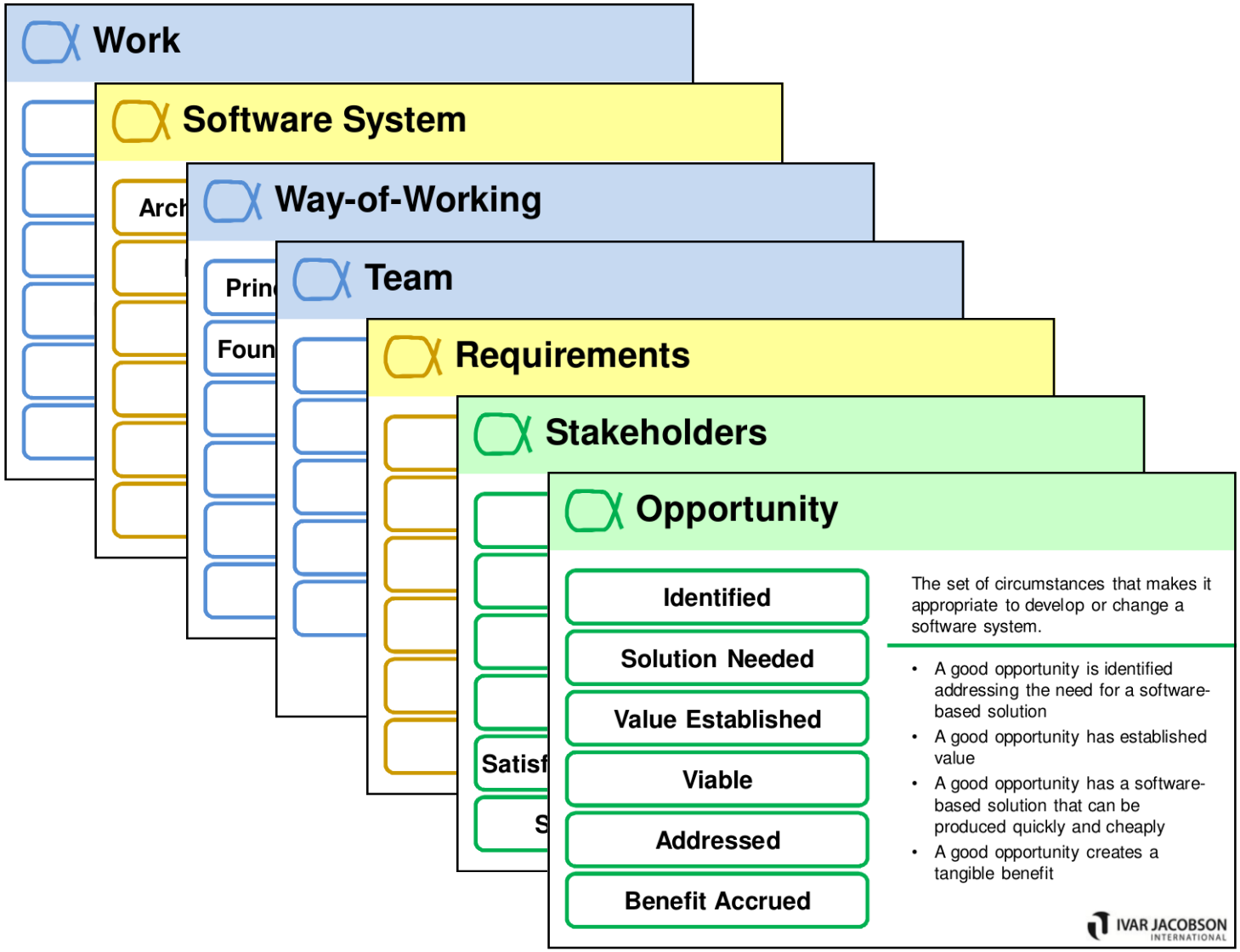
Verifikation:

Med *verifiering* menar man, något striktare, att slå fast att produkten (och steg på vägen!) lever upp till kravspecifikationen! Denna kan ju beskrivas av mer än funktionaliteten (andra kvaliteter). Verifieringen kan också avse koll av mer interna tekniska lösningar, allmänna programvarukrav, etc..

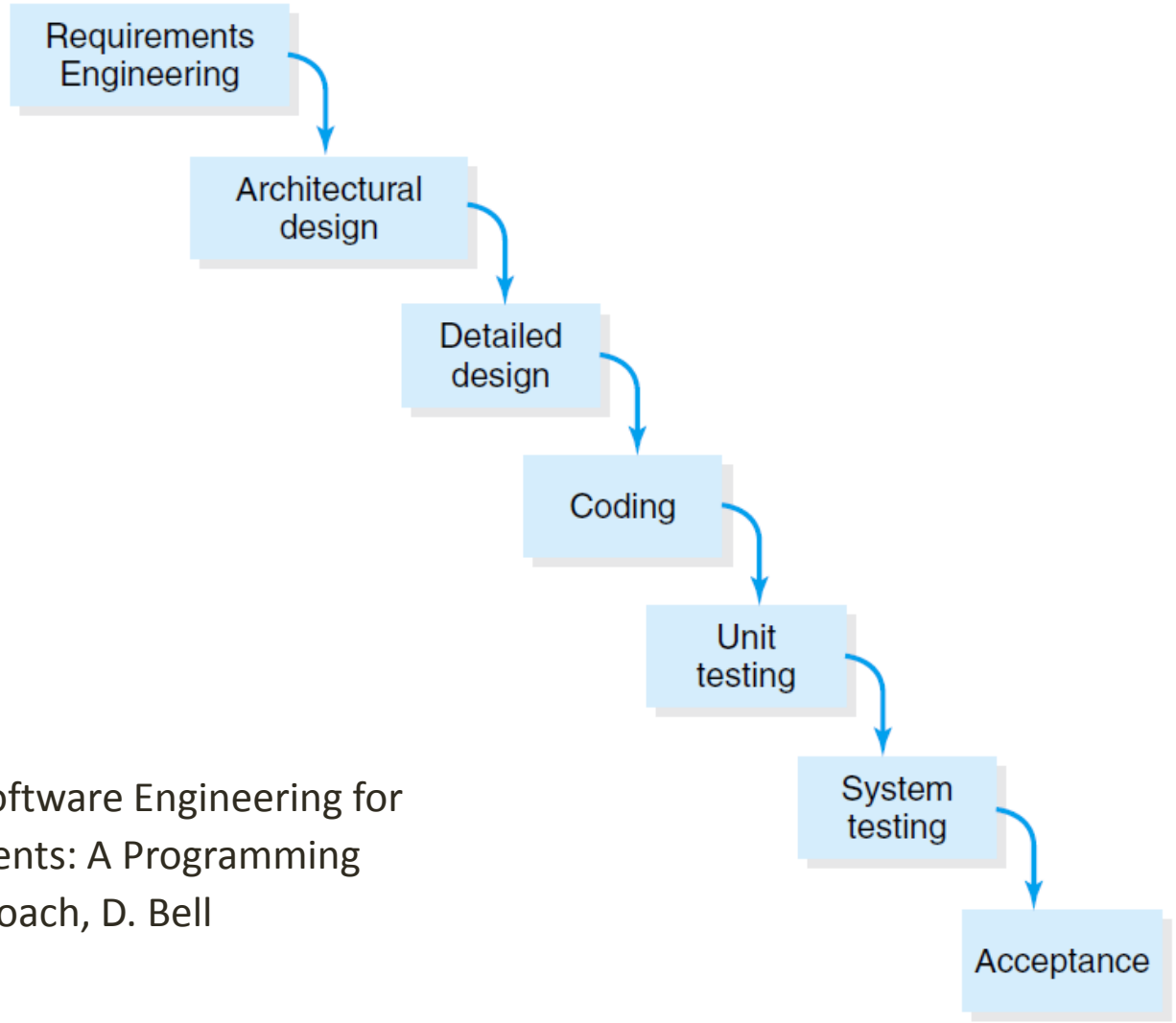
Lite populistiskt uttryckt (Boehm):

- Validation: Are we building the right product?
- Verification: Are we building the product the right way?

SEMAT Alpha Kernels

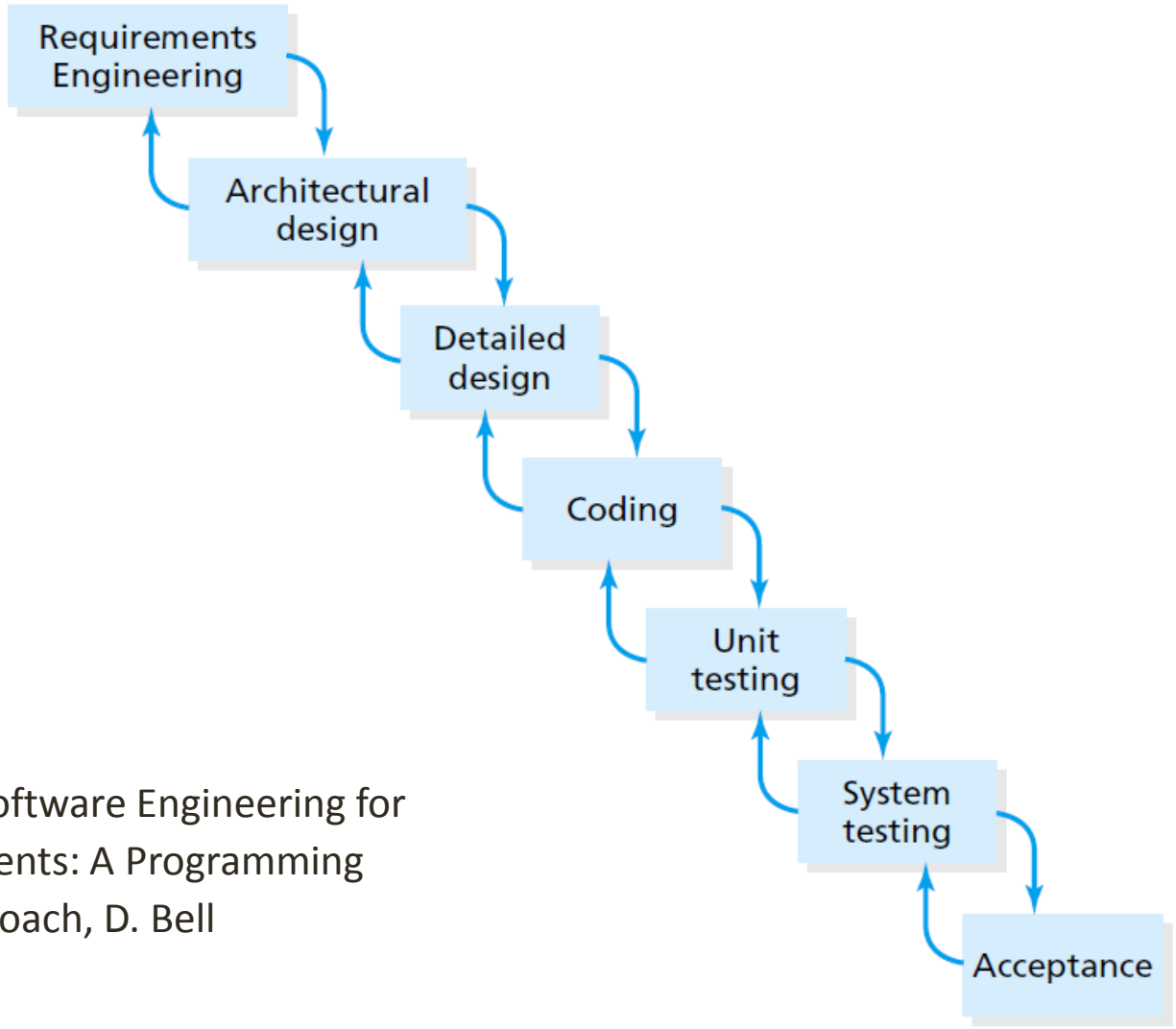


”Strikt” vattenfallsmodell



Källa: Software Engineering for Students: A Programming Approach, D. Bell

Vattenfallsmodell (”återhopp”)



Källa: Software Engineering for Students: A Programming Approach, D. Bell

Prototyping

Processmodell som bygger på att:

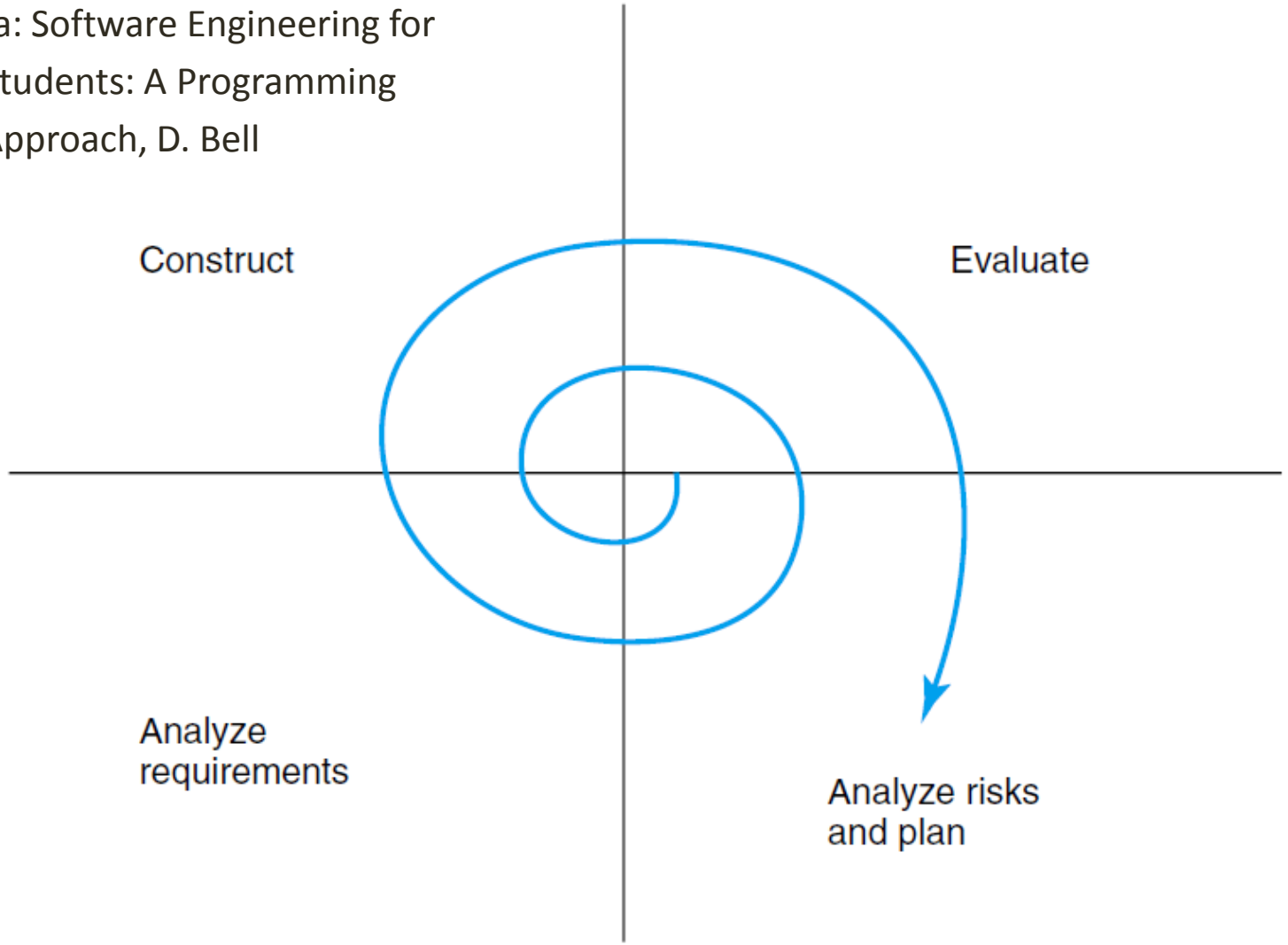
- Tidigt skriva en prototyp
- Låta användaren testa denna prototyp
- Förfina kravspecifikationen baserat på feedback
- Förfina prototypen och upprepa

Kan användas på två sätt:

- ”Evolutionary development”
  - Prototypen förfinas tills den utgör den färdiga produkten
- ”Throwaway prototyping”
  - Prototypen kasseras till slut
  - Den färdiga produkten utvecklas från grunden med tidigare lärdommar

Spiralmodellen

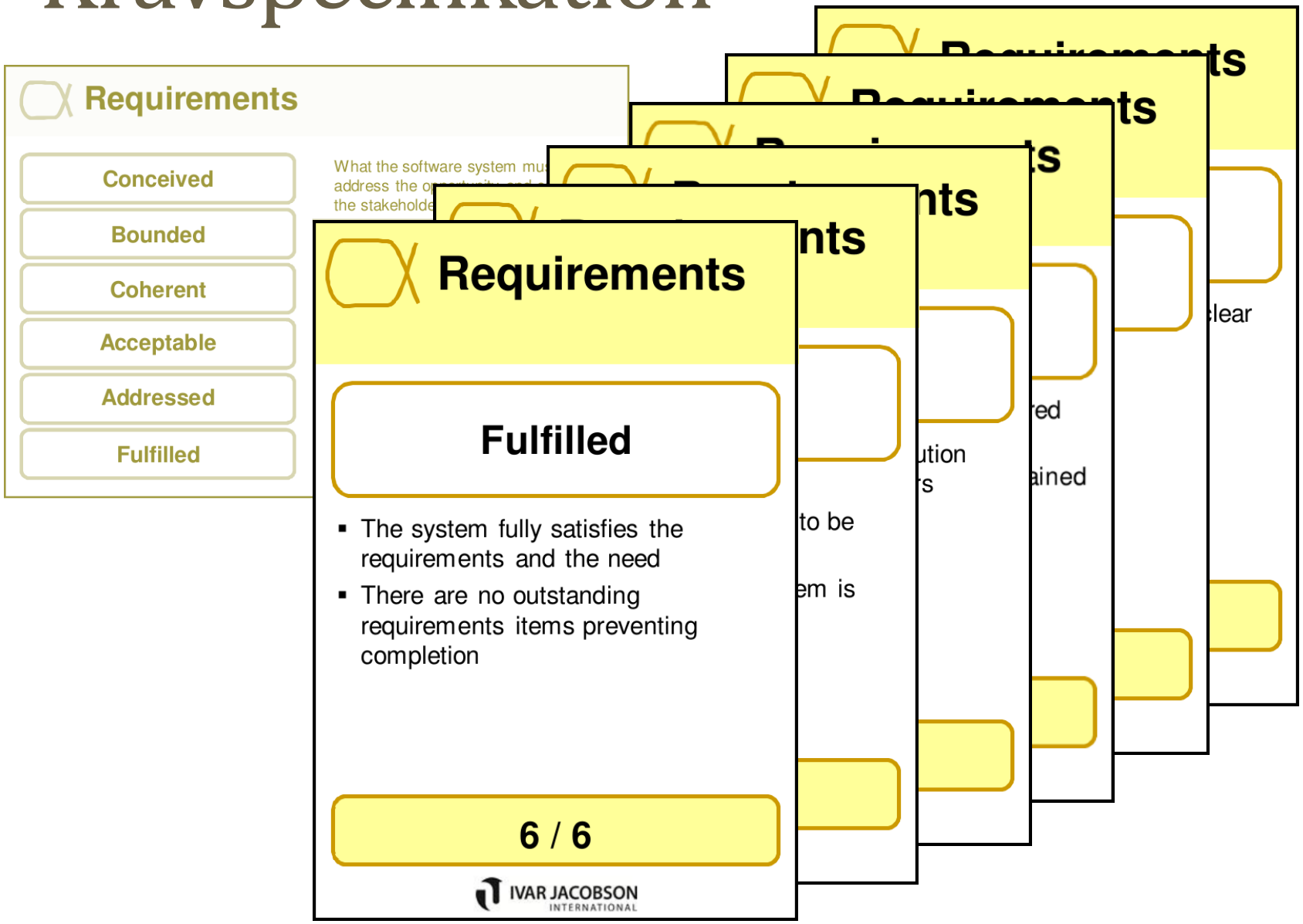
Källa: Software Engineering for Students: A Programming Approach, D. Bell



Tankar inför projektet

- Börja fundera på era principer och er metod
- Finns många fler att välja på än de som nämnts här
- Lägg upp en tidsplan med datum och vilka Alpha Kernel-tillstånd ni planerar att ha nått där. Använd också dessa för att övervaka var ni är och vad ni behöver arbeta på härnäst!
- Dokumentera era principer, er metod och er tidsplan i den rekommenderade projektplanen!

Kravspecifikation



En kravspec. kan innehålla..

- Identifikation
- Sammanfattning
- Innehållsförteckning
- Inledning
- Användarna
- Funktionella krav
  - Ska-krav
  - Bör-krav
  - ”Eventuellt”-krav
- Produktkomponenter
  - Programvara
  - Maskinvara
  - Dokumentation
- Effektivitet
- Kompabilitet
- Konfiguration
- Installation och service
- Tillförlitlighet
- Ordförklaringar
- Index

Exempel på ”verklig” kravspec.