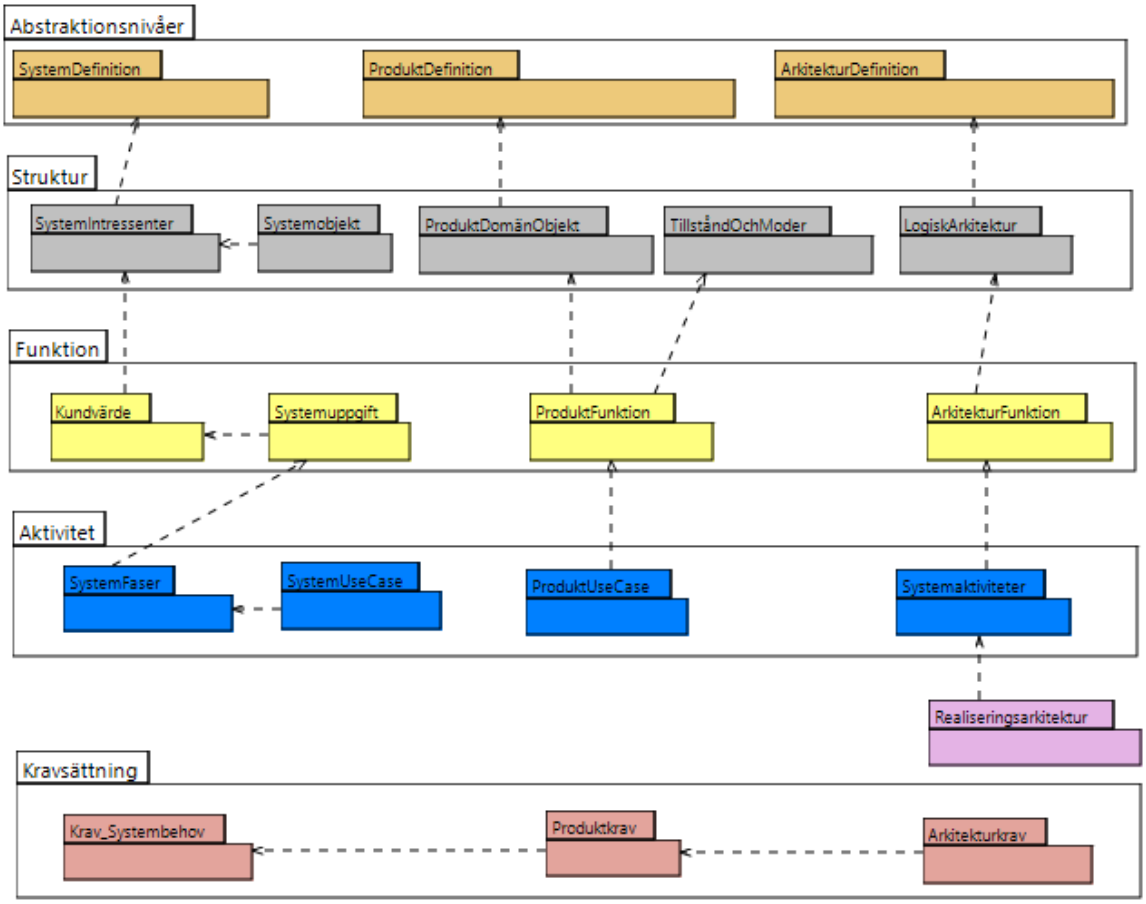
CHALMERS



PU²B-modellen

En introduktion till Model Based Systems Engineering (MBSE) utifrån användarcentrerad systemdesign

LARS-OLA BLIGÅRD
ROBERT NILSSON

Institutionen för produkt- och produktionsutveckling
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2015

PU^{2}B-modellen

En introduktion till Model Based Systems Engineering (MBSE) utifrån användarcentrerad systemdesign

LARS-OLA BLIGÅRD
ROBERT NILSSON

CHALMERS

Institutionen för produkt- och produktionsutveckling
CHALMERS TEKNISKA HÖGSKOLA

Göteborg, Sverige 2015

PU^{2}B-modellen
En introduktion till Model Based Systems Engineering (MBSE) utifrån användarcentrerad systemdesign

LARS-OLA BLIGÅRD
ROBERT NILSSON

Teknisk rapport vid Chalmers tekniska högskola
ISSN 1652-9243, rapport nr 91

Institutionen för produkt- och produktionsutveckling
Chalmers tekniska högskola
412 96 Göteborg
Telefon 031-772 1000
Göteborg, Sverige 2015

Omslag:
Strukturen för Product Utility Usability Business Model (PU^{2}B-modellen)

Tryckt av
Reproservice, Chalmers tekniska högskola
Göteborg, 2015

Förord

Idén till PU²B-modellen uppstod under en god måltid där författarna konstaterat att det trots mycket kunskap, saknas effektiva former för tillämpningar när det kommer till ämnesområdena Human Factors och Systems engineering. Då vi båda nu varit verksamma tillräckligt länge för att själva anse oss ha tagit del av både teoretiska och praktiska aspekter av ämnesområdena, kunde det konstateras att det visst fanns riktigt bra metoder för både teoretiker och praktiker men att det saknades en enkel brygga mellan de två världarna.

Denna rapport presenterar den nya PU²B-modellen, vilken är ett försök i att på ett enkelt sätt möjliggöra en start av modellering på ett sätt som ska vara lättillgänglig och användbart. PU²B-modellen är också tänkt att överbrygga traditionella områden som marknad, användarnytta, användbarhet och teknisk lösning.

Fokus för skrivande av rapporten har varit att skapa ett material som presenterar modellen i ett sammanhang och visar på hur den kan användas praktiskt. Detta fokus har medfört att exemplen kanske ur vissa perspektiv inte presenteras formellt korrekt i alla delar. Om du kommit så här lång borde resten gå lätt, vi hoppas du har nytta av PU²B-modellen. Självklart tar vi gärna feedback!

Lars-Ola Bligård, Civilingenjör Elektroteknik. Teknologie Doktor Människa - Teknik - Design. Verksam som forskare inom Human Factors Engineering vid Chalmers tekniska högskola.
lars-ola.bligard@chalmers.se

Robert Nilsson, Civilingenjör Maskinteknik, Teknisk Licentiat Sjöfart och Marin teknik. Har arbetat med Human Factors och Systems Engineering från systems helhetsperspektiv för Saab AB, Altran Technologies och numera Volvo Car Group.
robert.nilsson.2@volvocars.com

Innehåll

1 Inledning

Kapitlet ger en övergripande beskrivning av problemet och introducerar sedan PU²B-modellen som ett förslag på lösning.

1.1 Bakgrund

Komplexiteten avseende relationer både mellan och inom produkter förutspås fortsätta öka. En tydlig indikation för detta är Ericssons vision om ”Networked Society” [1] som visar en framtid där maskiner kommunicerar med varandra för att underlätta människors vardag. I det ”lilla” systemet för en enskild individ kan detta redan idag exemplifieras med möjligheten att via appar på distans kontrollera och styra exempelvis värmesystem och larm för hushåll. Ytterligare ett exempel på ökad komplexitet och nya samverkansformer mellan människan och dess maskiner är Volvos Self Driving Car[2]. En sådan produkt skulle fundamentalt ändra både våra förväntningar och användarmönster jämfört med hur fordon uppfattas och används idag.

Det har empiriskt visat sig att produkter som erbjuder nya möjligheter och hög tillgänglighet får hög genomslagskraft. Smartphones genomslag på marknaden har tydligt visat att produkter som erbjuder möjligheter och som upplevs ge tillgänglighet värderas högt och blir eftertraktade. Det har till och med debatterats att föräldrar till den grad använder Smartphones att det blivit en konkurrent för barns uppmärksamhet [3].
Författarnas uppfattning avseende leverantörer av Smartphones är att de hellre ser sig som leverantörer av smidiga och smarta lösningar för att föräldrar ska få mer tid för sina barn än att det blir verktyg för sociala medier som ”stjäl” föräldrar från sina barn. En central del i detta exempel är givetvis själva definitionen av vad ett system är och vem som bör ta ansvar för vad. Kanske ska inte leverantörer av mobiltelefoner över huvud taget anses bära ansvar för att de erbjuder möjligheter. Men, om frågeställningar av denna typ inte finns med strukturerat i koncept- och designarbetet, och hanteras som en del av förståelsen av produktens användarkontext och dess potentiella inverkan på de system de verkar i, finns det en risk att det slår tillbaka på produkten eller användarna.
En central utgångspunkt för behovet av en mer strukturerad systemdesign utifrån människans perspektiv är att utveckling av teknik går i snabb takt. Detta är en centralt på det sättet att författarna, efter mer än 10 års individuell erfarenhet av produktutveckling och forskning, menar att det är just möjligheterna och tekniken som traditionellt prioriterats. Då är frågan ”Vad kan vi göra med denna nya möjlighet” mer intressant än ”Varför skulle vi vilja använda denna nya möjlighet”. Detta fokus på möjligheter anses vara sprungen från den traditionella ”teknikerns” hängivna passion av att lösa problem och det anses därmed fundamentalt påverka utvecklingsprocesser, i båda stora och små företag. Konsekvensen är att det lätt blir de tekniska möjligheterna som får fokus och inte det faktiska behovet hos användaren, vilket då riskerar att bli sekundärt. Tyvärr riskerar detta perspektiv i det långa loppet att medföra problem, då produktens komplexitet medför att det kan bli svårare att hålla ihop produkten och dess utveckling, för att nå ett strategiskt mål.
Målet med rapporten är att presentera Product Utility Usability Buisness Model, förkortat PU²B-modellen. Syftet med modellen är tvådelat. Först, modellen är tänkt att fungera som ett verktyg för att systemdesigners snabbt och enkelt ska kunna börja modellera system. På några få timmar och med enkla hjälpmedel som penna och papper kan de första vyerna enkelt ta form.

Det andra syftet med modellen är att, inom ramen för Systems Engineering, använda de erfarenheter från forskning inom Human Factors där nödvändigheten att hantera komplexitet snabbt blivit ett måste på samma sätt som för mjukvaruutveckling. Genom att kombinera metoder från våra erfarenheter från Human Factors med Systems Engineering-perspektiv, möjliggörs en strukturerad modell för att hantera produkters komplexitet som kan svara upp mot framtidens behov.

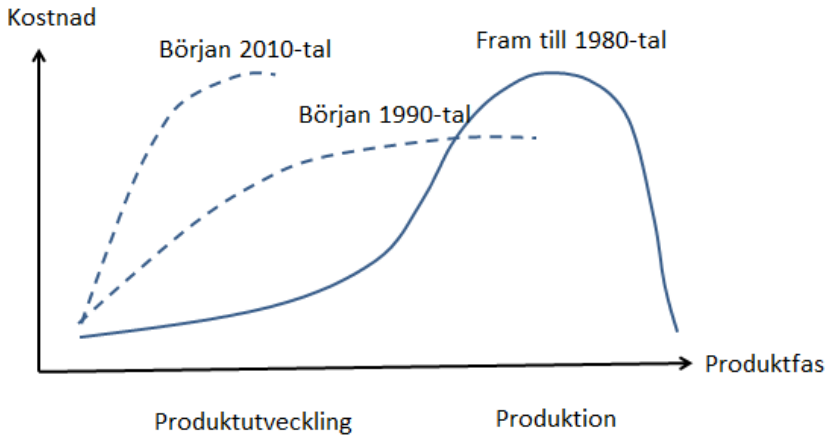
---

[1] http://www.ericsson.com/thinkingahead/networked_society [2013-12-31]
[2] http://www.nyteknik.se/nyheter/fordon_motor/bilar/article3791107.ece [2013-12-31]
[3] http://www.svd.se/nyheter/inrikes/forskolelarm-foraldrar-slapp-mobilen_8657440.svd [2013-12-31]

PU²B-modellen tar hänsyn till produktens marknadsperspektiv, tänkt användning och kundvärde samt tekniska förutsättningar. Genom att använda denna modell erbjuds möjlighet att omfamna framtidens möjligheter och att utnyttja dessa på bästa sätt genom att ta medvetna beslut i en föränderlig värld med både snabb teknikutveckling och ökad komplexitet.

1.2 Problematisering

Problemställningen i rapporten baseras bland annat på en generalisering från en avhandling av Wickelgren (2005), som varit verksam inom forskning knuten till bilindustrin. Wickelgren menar i sin avhandling att det funnits två perspektiv på vad som ger vinst. Det ena perspektivet är produktionsinriktat och där arbetas det med inriktning på att skära kostnader i produktion, traditionell ”Lean Production”. Det andra perspektivet är intäktsorienterat och enligt det får varumärkesstrategi och produktens upplevelse stort fokus. Med detta följer då naturligt en ökad kostnad för produktutveckling för att kunna kombinera både varumärkesstrategi och utvecklingsarbetet. Förändringen blev enligt Wickelgren, extra påtagligt vid millennieskiftet. Figur 1.1 nedan visar en tolkning på hur kostnader skulle kunna fördelas mellan produktutveckling och produktion under de fyra närmsta årtionden. Samtidigt som produktutvecklingsaktiviteter från och med 80-talet krävde fler arbetare försökte företagen skära i bemanning för produktion.



Figur 1.1 Kostnadsförskjutning bildindustrin över tiden

Ökade kostnader för produktutveckling förklaras, inom bilindustrin, bland annat av ökad internationell konkurrens, produktfragmentering på grund av kunders diversifierade krav och behov samt den ökade komplexiteten för produkter. Den sistnämnde förklaringen kan exemplifieras med att fler delar i bilen blivit elektriskt styrda och att tidigare rent elektriska eller rent mekaniska komponenter har kopplats samman. Antalet ECU:er (Electronic Control Unit) har ökat dramatiskt och med dem också komplexiteten rörande deras relationer och gränssnitt.

Ett sätt att utveckla resonemanget vidare är att beakta de tankar kring ”The internet of things” som nu växer fram. Det kommer att ställa ökade krav och medföra högre kostnader i samband utvecklingsarbetet. Författarna anser att både faktorerna rörande komplexitet och ”the internet of things” är generaliserbara till andra industrier än bilindustrin. Vi tycks i så fall vara mitt uppe i ett paradigmskifte mot nya affärsmodeller och tjänster, i molnet, samtidigt som produktutveckling redan idag har fullt upp med att lösa de problem som redan finns, bland annat på grund av den redan upplevda förhöjda komplexiteten.

Inom vår yrkesroll som Human Factors ingenjörer har vi arbetat med denna typ av frågeställningar. Det är just i mötet mellan tekniken och människan som komplexiteten blir uppenbar och att den inte kan gömmas i ett ytterligare lager av teknik. Mål och syfte för slutkund har alltid behövt vara i fokus för framgångsrik interaktion. Det är alltså tydligt att det är nödvändigt att hantera en ökad komplexitet på ett strukturerat sätt, i strävan att nå önskad effekt och kontroll under både nyutveckling och fortsatt utveckling av produkter. Kort sagt, vi måste arbeta smartare och det är en av grunderna för att både Systems Enginering och Human Factors Engineering kommer att behövas mer.

De svårigheter och svagheter som rapporten främst försöker adressera med PU²B-modellen är:

- Svårigheter att beskriva en produkts kontext på ett sådant sätt att det underlättar förståelsen för produktens mål och syfte, vilket är en förutsättning för att koppla ihop marknadsverksamhet med teknisk utveckling.
- Svårighet att tydligt beskriva en produkts mål och syfte samt koppla detta till värde för slutkund, samtidigt som dessa bryts ner dessa på ett tydligt sätt för de olika instanserna inom produktutveckling.
- Svårighet för industrin att tydligt kunna hantera utveckling som både härrör från teknisk utveckling och från användares behov. Vid traditionell produktutveckling, ”bottom up”, baseras ett tekniksteg på vad som går att göra och inte utifrån frågan varför det ska göras för slutanvändaren. Det kan vara värt att betrakta tekniksteg från användarperspektiv och ”top down” med.
- Svårigheten att resonera utifrån ”ren” funktionalitet, vilket är något som ingenjörer traditionellt har svårt för. Det bör därför lyftas fram specifikt. Om en produkts funktionalitet inte frikopplas från lösning blir det svårt för företaget att hålla en långsiktig strategi med snabba och flexibla möjligheter att integrera ny teknik.

1.3 Förslag på lösning - PU²B-modellen

Produktutvecklingsprocesser tenderar, som övrig industri, att präglas mer och mer av tänk utifrån ”Lean”. Detta betyder att det aktivt arbetas med att reducera de aktiviteter som inte medför något värde. Därför är medvetna beslut och möjlighet att följa upp medvetna val av central betydelse för företagets utveckling.
Startpunkten för PU²B-modellen finns i Systems Engineering, närmare bestämt från INCOSE handbook of Systems Engineering (Haskins, 2011) och den övergripande struktur som ISO 15288 (ISO, 2008) erbjuder. ISO 152888 bygger på fyra grupper av processer: Avtalsprocess (Agreeement processes), Organisationsprocess (Organistaional project-enabeling processes), Projektprocess (Projectprocesses) och Tekniska processer (Technical processes). Av dessa fyra riktar sig PU²B-modellen främst mot tekniska processer och ytterligare mer specifikt mot den första halvan av dessa, nämligen själva produktutvecklingen. För att kunna hantera frågeställningar har metoder och verktyg från området Model Based Systems Engineering (MBSE) ansetts vara särskilt kraftfulla.

Den stora nyttan med PU²B-modellen kan sammanfattas med att den erbjuder en tydlig struktur för att koppla ihop kundcentrerat arbete med teknikcentrerat arbete, genom ett fokus på funktion och aktivitet som en länk däremellan. PU²B-modellen erbjuder en struktur för uppföljning och ställningstagande utifrån marknadsperspektiv och medger därmed medvetna beslut under produktutveckling.

Grundtanken med PU²B-modellen är att bygga upp en systemmodell med hjälp av enkla typer av objekt som adderas och successivt kombineras till en mer heltäckande modell. Genom att särskilja typer av objekt visas det tydligt vad ett system består av (struktur), vilka förmågor ett system behöver ha (funktioner) och hur dessa skapar kundvärdet (aktiviteter) samt hur produkten ska kunna produceras (realisering). Som ett stöd för utveckling erbjuder PU²B-modellen också tydlig en koppling mot krav.

Modellering av ett system sker med hjälp av fem typer av modellobjekt:

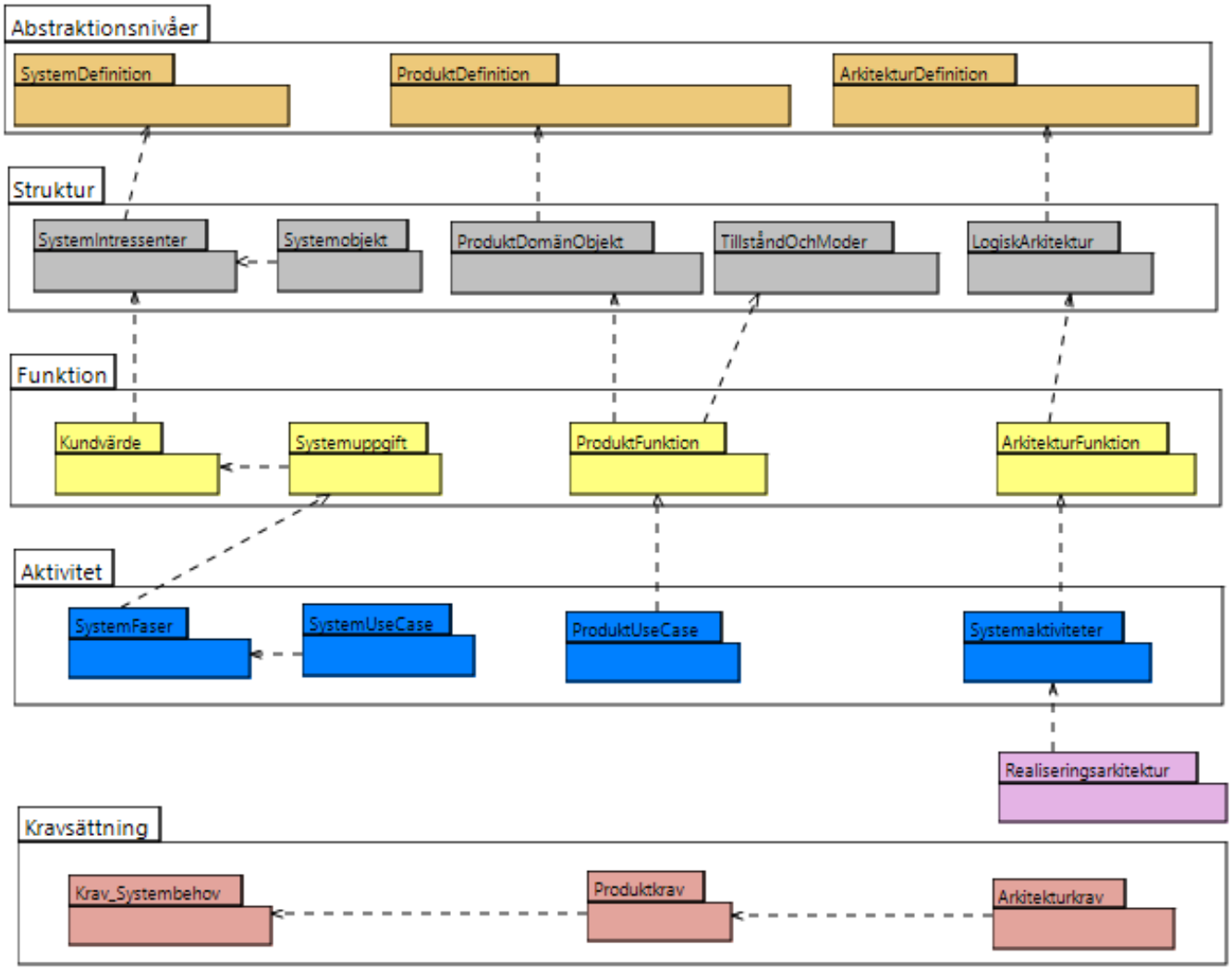
- Struktur: Vilka strukturelementet som ingår i ett system för att det ska kunna analyseras och hanteras.
- Funktion: Hur elementen påverkar varandra och ändrar systemets tillstånd.
- Aktiviteter: Hur systemet uppnår målen och samverkar med sin kontext.
- Realisering: Hur produkten i systemet fysiskt existerar.
- Kravsättning: Vad som är väsentligt att beakta för att kunna hantera produktfrågor på olika abstraktionsnivåer (se nedan) och ge stöd för fortsatt utveckling.

Ovan beskrivna typer av modellobjekt återkommer typiskt i olika former under utveckling av en produkt. De kan vara av liknande karaktär samtidigt som de skiljer sig åt genom att beskriva olika abstraktionsnivåer av den kommande produkten. Precis som det finns systemkrav för hela system och delsystemkrav som bara berör specifika delar av ett system, finns det behov av att resonera kring de olika typerna av objekt på olika abstraktionsnivåer.

För att kunna hantera frågeställningar med rätt objekttyp i sitt mest lämpade sammanhang har det därför skapats tre abstraktionsnivåer för ett system:

- Systemdefinition: fokus på kundvärde och användarbehov samt gränssnitt mellan systemet och dess omvärld (systemet betraktas som ”Black box”)
- Produktdefinition: fokus på användning samt systemets huvudbeståndsdelar och systemets övergripande interna gränssnitt (systemet som ”Grey box”)
- Arkitekturdefinition: fokus på de tekniska principerna för komponenter och interna gränssnitt ska tydliggöras (systemet som ”White-box”)

I Figur 1.2 återfinns en övergripande presentation av abstraktionsnivåer samt de typer av modellobjekt och dess representationer som anses behövas för att framgångsrikt kunna modellera ett system.



Figur 1.2 PU²B-modellen övergripande

De tre abstraktionsnivåerna visar tydligt hur systemmodelleringen går från marknadsorienterade aktiviteter till teknisk utveckling och realisering, via användningen. Denna rapport kommer att fokusera mest på de två första abstraktionsnivåerna för att det anses finnas störst behov och bli störst bidrag att visa en struktur för dessa. Det är också inom dessa abstraktionsnivåer relativt enkelt att generalisera och arbeta med enklare verktyg (som Powerpoint eller Visio). Många organisationer har redan väl fungerade processer för utveckling av sina produkter med avseende på processer, verktyg, Configuration Management mm. När arbetet når abstraktionsnivån för arkitektur (med realisering) blir situationen för den enskilda organisationen av stor betydelse och därför bör PU²B-modellen särskilt anpassas till organisationens egna processer.

1.4 Läsanvisning

Rapporten fokuserar på användning av PU²B-modellen i uppstart och de tidiga faserna av ett utvecklingsprojekt och vänder sig därför till olika läsargrupper med olika syften (tabell 2.1).

Tabell 2.1 Målgrupper för rapporten

Målgrupp	**Syfte**
Utvecklingschefer	Stöd för planering genom att visa vad som är viktigt att fånga inom respektive abstraktionsnivå. Detta förväntas underlätta bedömning av utvecklingsprocess samt resurs och kompetensbehov
Projektledare utveckling	Stöd för hur ett projekt ska organiseras för att täcka rätt frågeställningar utifrån arbetet i respektive abstraktionsnivå
Systemingenjörer	Öka förståelse för hur marknads- och användaraspekter relaterar till teknisk arkitektur
Human Factors - ingenjörer	Stöd för hur HF-aspekterna integreras i utvecklingsprojektet
Marknadsanalytiker	Ökad förståelse för hur kundvärden kan förädlas och integreras i utvecklingsprojektet

Det följande innehållet i rapporten är indelat i sex kapitel med olika innehåll och syften:

Kapitel 2: Här beskrivs de teoretiska ramverk som PU²B-modellen vilar på. Kapitlet är lämpligt för de som vill få en djupare förståelse för de områden där modellen verkar inom och har hämtat sin inspiration från.

Kapitel 3: Kapitlet beskriver PU²B uppbyggnad i stort och de centrala modellelementen som behövs. Kapitlet är lämpligt för de läsarna som vill förstå hur modellen är fungerar mer i detalj.

Kapitel 4-6: Dessa kapitel beskriver de tre abstraktionsnivåerna i modellen än mer i detalj och exemplifierar dem illustrativt. Kapitlen vänder sig till de läsare som är intresserade av att praktiskt applicera PU²B-modellen.

Kapitel 7: Här behandlas slutligen hur PU²B-modellen relaterar till existerande arbetssätt. Kapitlet vänder sig till dem som vill leda och organisera PU²B-modellen i en verksamhet.

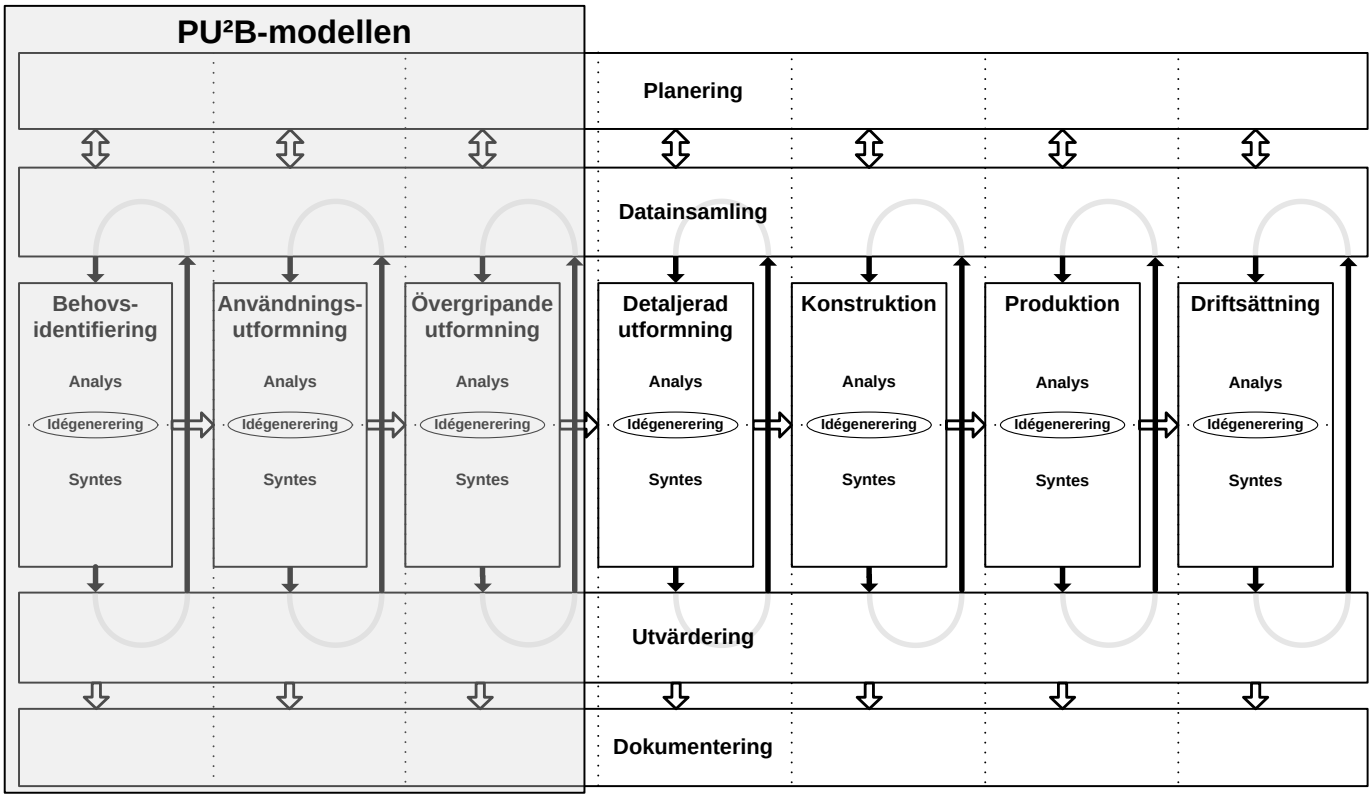
2 Teoretiska ramverk

Kapitlet beskrivs de teoretiska ramverk som PU²B-modellen vilar på. Inledningsvis relateras till produktutveckling och sedan följer Systemteori och Systems Engineering (med systemmodellering och MBSE). Nästa ramverk är Human Factors och Human Factors Engineering och kapitlet avslutas med beskrivning av funktion och aktivitet utifrån aktivitetsteori.

2.1 Produktutveckling

Produktutveckling innefattar allt det arbete som behövs för att få en ny produkt realiserad. Enligt Nationalencyklopedin är produktutveckling *"...det förlopp som föregår framtagandet av en ny produkt i ett företag eller en organisation. Processen spänner från brainstorming, produktutformning, beaktande av miljötålighetsteknik och marknadsundersökningar till konstruktion, tillverkning och marknadsföring.*" Produkt ska här ses som ett mycket brett begrepp som både kan avse artefakter och tjänster, samt kombinationer av dem. Mycket av utmaningen i produktutvecklingen är att hitta en balans mellan utformning, teknik, ekonomi, produktion, miljöpåverkan mm.

För att förenkla utvecklingsarbetet är ofta produktutvecklingsprocesser uppdelade i faser. Uppdelningen beror på skiftande innehåll och mål i och med utvecklingsarbetet fortskridande. Det finns många varianter på hur en produktutvecklingsprocess kan delas upp och den uppdelning som denna rapport utgår ifrån, kommer från ACD[3]-processen[4] (Bligård, 2015). Att just denna uppdelning valts beror på att ACD[3]-processen har en fokus på de tidigare delarna av utvecklings processen samt en tydlig koppling Human Factors, vilka båda är relevanta för PU²B-modellen.



Figur 2.1 ACD[3]-processen med relevanta delar för PU²B-modellen markerade

ACD[3]-processen är uppdelad i sju vertikala faser, var av PU²B-modellen berör främst tre första, se figur 2.1.

[4] ACD[3] står för Aktivitets Centrerad Design och trean syftar på processen tre strukturer: vertikal, horisontell och inre. ACD[3]-processens inre struktur utvecklades utifrån PU²B-modellen.

1. Behovsidentifiering

Syfte: undersöka hur omgivningen inverkar på den kommande lösningen och hur lösningen ska påverka omgivningen, samt vad användaren har för behov och värderar i en lösning

Mål: Utforma den effekt som lösningen ska ha på det sociotekniska systemet och välja princip för användningen

2. Användningsutformning

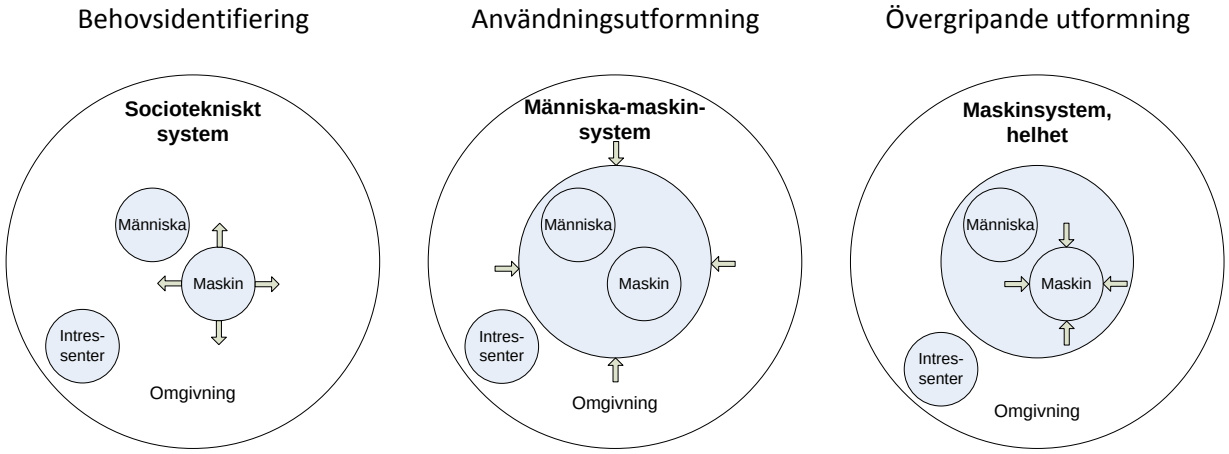
Syfte: Undersöka vilken användning som uppfyller behoven och ger avsedda effekter och undersöka vilka övergripande (tekniska) lösningar som uppfyller användningen

Mål: Utforma användningen och välja teknisk lösningsprincip

3. Övergripande utformning

Syfte: Undersöka vilken teknisk uppbyggnad av maskinen som ger avsedda effekter och undersöka hur samspelet mellan människan och maskinen bör ske

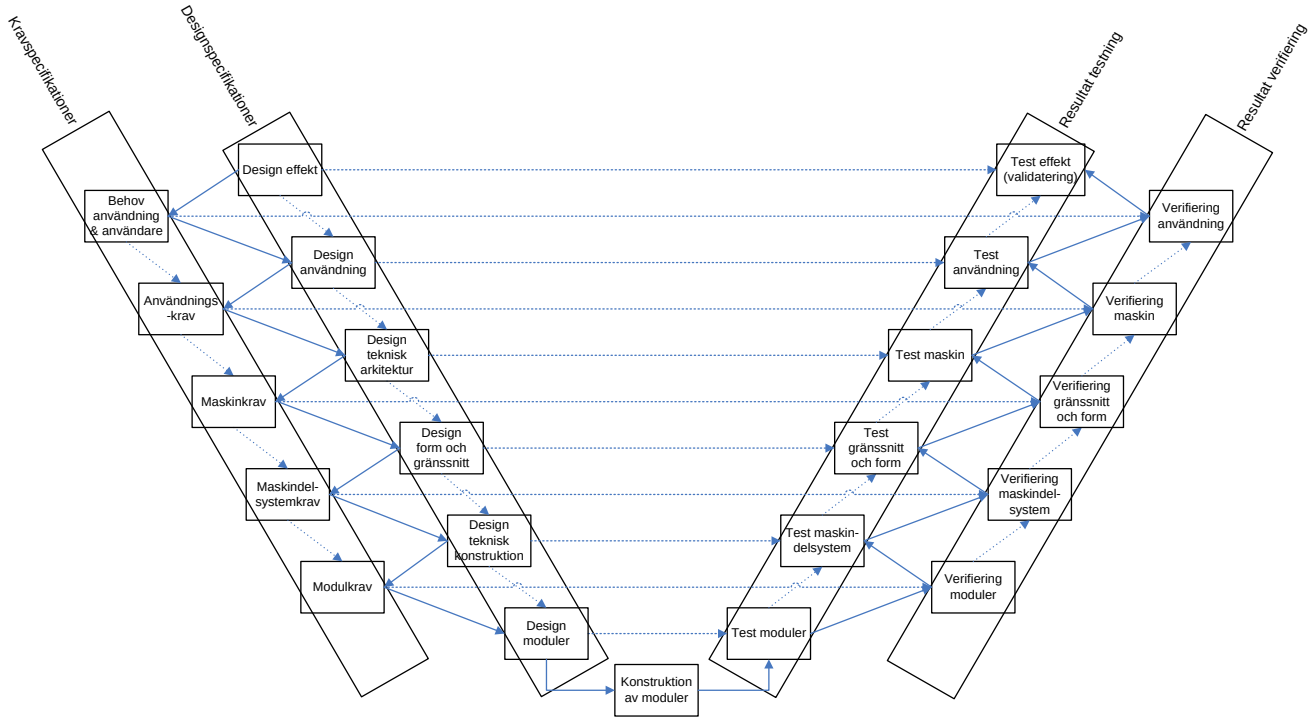
Mål: Utforma tekniska arkitektur och välja princip för interaktion, estetik och form



Figur 2.2 Perspektiven för de tre första stegen i ACD³-processen

Relevant för PU²B-modellen är också de olika perspektiv som finns i de tre första faserna, se figur 2.2. Under behovsidentifieringen är perspektivet omgivningen betraktad från den maskin som ska utvecklas och här blir arbetet naturligt användarcentrerat (jmf figur 1.1, Systemdefinition). Under användningsutformningen är perspektivet människa-maskinsystemet, betraktat från omgivningen och här blir arbetet naturligt användningscentrerat (jmf figur 1.1, Produktdefinition). I den övergripande utformningen är perspektivet maskinen i helhet, betraktad från omgivningen och här blir arbetssättet naturligt teknikcentrerat (jmf figur 1.1, Arkitekturdefinition). De olika perspektiven kommer i från de systemen som behöver beaktas: sociotekniskt system, människa-maskinsystem respektive maskinsystem. Tanken med perspektivskiftet att hjälpa processen att byta fokus under utvecklingsarbetet och därmed undvika att någon viktig aspekt missas genom att flera systemsynsätt gås igenom.

Under hela produktutvecklingsarbetet finns två parallella informationsflöden som styr arbetet: krav och design. Kravsättningen har som mål att avgränsa designrymden, medan designarbetet ger en lösning inom avgränsningen. Under utvecklingsarbetet sker sålunda en successiv växelverkan mellan design och krav när de olika faserna löps igenom. Ett illustrerande exempel (figur 2.3) kommer från ACD³-processen, vilken visar på den dubbla V-modellen där design och krav är de två vänstra pelarna. Designen i en fas är utgångspunkter för kraven i den fasen. Kraven sätter sedan ramarna för designen i nästkommande fas, och så vidare. De två vänstra pelarna är på samma sätt delade i testning och verifiering, där testning är utvärdering mot den specificerade designen, medan verifiering är testning mot de specificerade kraven.



Figur 2.3 Dubbla V-modellen från ACD³-processen (Bligård 2015)

2.2 Systemteori och Systems Engineering

PU²B-modellen har fått sin grundläggande struktur från tankar inom Systems Engineering (SE), närmare bestämt inom Model Based Systems Engineering (MBSE). Båda vilar i sin tur på systemteori. Systemteori är ett samlingsnamn för de teorier som används för att beskriva hur delar tillsammans formar en helhet med andra egenskaper än de enskilda ingående delarnas (Flood och Carson, 1993; Skyttner, 2005). Systemteori utgår ifrån att det inte går att förstå en helhet enbart genom att bryta ner den i mindre delar och därefter studera delarna separat. Det går exempelvis inte att förklara hur en människa fungerar genom att enbart studera människans celler. Systemteori fokuserar alltså på ordningen och relationen mellan ingående delar som förenar dem till en helhet. Synsättet kommer ursprungligen från biologin, men används nu vida spritt inom både naturvetenskap, teknik, psykologi och samhällsvetenskap. Systemteorins angreppssätt gör att den används för att beskriva och förstå systems komplexitet.

I systemteori är just själva systemet det centrala begreppet. Ett system består av flera kommunicerande element i en organiserad helhet. Den organiserade helheten kan vara verklig, som till exempel en maskin, eller abstrakt, som till exempel regler. Ett element kan på samma sätt vara något fysiskt, socialt eller abstrakt. Kommunikationen mellan elementen kan utgöras av överföring av materia, information eller energi/kraft.

Varje system har en systemgräns som definierar vad som tillhör systemet och vad som inte tillhör systemet. Gränsen kan vara fysisk, som för en maskin eller ett djur, social som för en flock, eller abstrakt, som för en regelsamling. Ett system kan i sig vara en del i ett större system och på samma sätt kan ett element vara ett system i sig och sin tur vara uppdelad i mindre element. Syftet med varje system är att processa energi, information eller materia till ett resultat för att användas inom systemet, utanför systemet (omgivningen) eller bådadera.

Vad säger att något är ett system? För att kunna säga att ett system finns, behöver oftast följande karaktäristik finnas (Flood och Carson, 1993; Skyttner, 2005):

- Helheten är mer än summan av delarna.
- Helheten definierar naturen för delarna.
- Delarna kan inte förstås genom att studera helheten.
- Delarna är dynamiskt relaterade och beroende av varandra.

En grundläggande idé inom systemteori är holism, dvs att helheten är mer än summan av de ingående delarna. Det innebär att ett system som helhet fungerar annorlunda än elementen i systemet och att elementen inte enskilt kan göra vad systemet som helt kan. Finns inte denna holism, så finns det heller inget system.

Ett element anses tillhöra ett system om det finns inom systemgränsen och elementet har en relation (kommunicerar) med andra element innanför systemgränsen. Viktigt för systemteori är därför att beskriva det flödet av energi, materia och information inom och över systemgränsen, samt att beskriva de olika elementens relation och hur de påverkar varandra.

Systems Engineering (SE) är ett tvärvetenskapligt ingenjörsämne som bygger på systemteori och fokuserar på hur komplexa tekniska system kan utformas och hantera under hela sin livscykel. Aspekter, såsom tillförlitlighet, logistik, samordning, mätningar, utvärderingar och med flera blir svårare vid arbete med stora eller komplexa projekt. SE kan beskrivas som en sätt att arbeta holistiskt med en produkt från flera olika perspektiv för att överkomma svårigheten med komplexitet. SE innefattar därför bland annat arbetsprocesser, optimeringsmetoder och verktyg för riskhantering. Innehållet i SE överlappar därför naturligt med andra discipliner inom teknik och organisation så som reglerteknik, industriell ekonomi och projektledning.

Avsikten med SE är att garantera att alla relevanta aspekter av ett projekt eller system beaktas och integreras till en helhet. Alla användaraspekter längs livtidscykeln för en produkt behöver omhändertas och därför är Human Factors en viktig del inom Systems Engineering och benämns där oftast som Human System Integration. Processer från SE är typiskt ”top down” orienterade, vilket stämmer bra med inledningskapitlets resonemang om att den ökade komplexiteten kommer ställa krav på ökade resurser i uppstarten av projekt just för att hantera ökningen av komplexitet.

2.3 Systemmodellering och MBSE

Ett av tillämpningsområdena inom Systems Engineering är Model-Based Systems Engineering (MBSE), vilket defineras av INCOSE (INCOSE, 2007) som ”*the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases*.” Målet med MBSE är att ersätta en utvecklingsprocess som är dokumentcentrerad, med en serie integrerade modeller som sträcker sig genom hela utvecklingsprocessen. Avsikten är att ge bättre insyn i utformningen av det tekniska systemet och en effektivare hantering av utvecklingsprocessen.
Det finns, redan idag, många områden där MBSE tillämpas och där det modelleras på ett eller annat sätt. Ett exempel på en domän där det länge modellerats är mjukvaruarkitektur. Ett annat exempel på domän där det modellerats är försvarssektorn, ett tydligt exempel på det är alla de ramverk som härstammar därifrån. Dess behov kan komma från komplexiteten i att samordna stora insatser där effektiv samverkan är särskilt viktig. Av dessa exempel framgår det tydligt att användningsområdena kan skilja sig mycket åt. Precis som det kan finnas olika mål och syfte finns det flera modellspråk, ramverk och verktyg som kan användas. Med PU²B-modellen vill vi inte argumentera för eller emot någon annan metod, ramverk eller förespråka något specifikt verktyg. Vi ser PU²B-modellen som en kombination av andra metoder och ramverk. Främsta syfte är att den på ett enkelt sätt och på kort tid ska erbjuda bra förutsättningar för systemdesign.

Ett modellspråk använder vyer eller typer av diagram för att representera ett system, eller delar av det. Modellspråket definierar hur olika typer av diagram kan se ut, vad de innehåller och ger exempel på vad de kan användas till. Ett modellspråk behöver dock inte peka ut hur dessa diagram entydigt ska kopplas. UML (Unified Model Language) och SysML (specialdel av UML) är exempel på modellspråk som kan användas för MBSE.

Ramverk kan användas som ett komplement till ett modellspråk genom att det specificerar vilka diagram som kan användas för att representera ett system eller en del av det. Ett ramverk syftar till att det enkelt ska gå att

jämföra och integrera olika system, eftersom system beskrivna enligt ett ramverk förväntas vara beskrivna på samma sätt. För den som är intresserad finns det ett flertal ramverk för hur modeller av system kan byggas. Några exempel är NATO Architecture Framework (NAF), Ministry of Defence Architecture Framework (MODAF), Department of Defence Architecture Framework (DODAF) eller The Open Group Architecture Framework (TOGAF).

Det finns ett antal metoder för att modellera system. En av de mest framträdande är HARMONY av IBM vilken bygger på UML/SysML. Denna metod kopplar olika diagram enligt modellspråken UML/SysML för att representera ett system. En metod är mer vag än ett ramverk på det sätt att det inte pekar ut färdiga vyer utan mer föreslår hur en modellarkitektur kan byggas upp för att vara tydlig i syfte att till exempel kunna generera mjukvarukod. Modellarkitekturen (metamodellen) för PU²B-modellen är empiriskt utvecklad utifrån erfarenhet av arbete med både Human Factors och Systems Enginering. Det är viktigt att poängtera att existerande modellspråk, ramverk eller metoder inte på något sätt betraktas som konkurrenter utan de har snarare varit en av de grundläggande förutsättningarna för att PU²B-modellen ska ha kunnat utvecklas. Genom att använda PU²B-modellen kan man bygga en bra grund för att gå vidare användning av mer detaljerade och omfattande metoder och ramverk.

2.4 Humans Factors och Human Factors Engineering

Nästa utgångspunkt för PU²B-modellen är Human Factors, även kallat ergonomi. Ämnet definieras av International Ergonomic Society (IEA, 2014) som ”*Ergonomics (or human factors) is the scientific discipline concerned with the understanding of interactions among humans and other elements of a system, and the profession that applies theory, principles, data and methods to design in order to optimize human well-being and overall system performance*.” Human Factors innefattar alltså att både optimera människans välbefinnande och att optimera övergripande systemprestanda, samt att det utgår från en systemsyn och med tydligt designfokus. Det finns olika områden inom Human Factors och den som är relevant för modellen är den ingenjörsmässiga applikationen kallad **Human Factors Engineering** (HFE). En väl använd definition av inom detta område är gjord av Chapanis (1985) som skrev att HFE “... *applies information about human abilities, limitations, and other characteristics to the design of tools, machines, systems, tasks, jobs, and environments for safe, comfortable and effective human use.*”. HFE innebär alltså att utforma och konstruera maskiner som är anpassade efter människans förmågor och som samtidigt är både effektiva och produktiva. Fokus ligger på det ingenjörsmässiga skapandet. Förenklat kan det sägas att HFE handlar om hur människan som användare bör integreras i produktutvecklingsprocessen.

För PU²B-modellen har Human Factors bidragit till ett fokus på hur användaren samspelar med en produkt för att uppnå ett mål. Inom Systems Engineering används ofta begreppet Human Systems Integration vilket enligt INCOSES handbok för Systems Engineering är en av de bärande pelarna för framgångsrik produktutveckling. Genom PU²B-modellen erbjuds ett ramverk där Human Factors (HF) kan integreras i systemmodellering:

- HF behandlar hur den mänskliga komponenten ska beaktas i systemet. Alltså att människan är en del i systemet och att de människliga komponenterna och tekniska komponenterna ska ses som ett system tillsammans och att deras samverkan behöver beaktas.
- HF har kunskap om hur människor fungerar i system och i samverkan med tekniska delsystem.
- HF har kunskap och metoder för hur människan ska beaktas på ett ingenjörsmässigt sätt under utvecklingsarbetet
- HF har kunskap och metoder för hur man testar och utvärderar hela systemet från det mänskliga perspektivet

Vidare stödjer tankarna inom Systems Engineering att Human Factors kommer till sin rätt genom att:

- åstadkomma en systemsyn genom ett utvecklingsprojekt
- kunna få ut de mänskliga aspekterna till respektive teknikgren eller delsystem

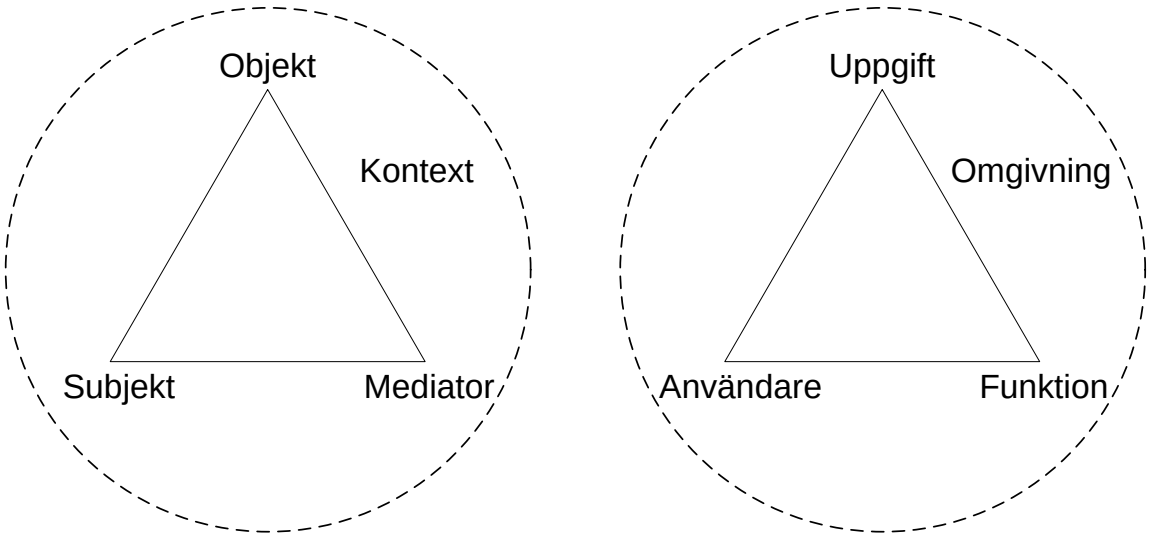
2.5 Funktion och aktivitet

För PU²B-modellen är det som systemet utför centralt, vilket redogörs genom att tydligt beskriva funktion och aktivitet. En **funktion** är en inbyggd förmåga hos ett system som relaterar något till någonting annat och definieras som en process av transformation eller ekvilibrium. En funktion har så att säga ett mål men inget syfte, och tid/rum är där med inte relevant i beskrivningen av ett system.

En **aktivitet** uppstår när en eller flera funktioner relateras till i en uppgift i tid och rum (och där med får ett syfte). Ett ramverk för att beskriva detta är aktivitetsteori (även kallad verksamhetsteori). Aktivitetsteori försöker förklara hur individer interagerar med sin omgivning och med artefakter. Grundtanken för användning av aktivitetsteori är att vi inte kan studera maskiner som enskilda ting, utan att vi måste studera hur de medierar användning.

Det finns många varianter av aktivitetsteori alltsedan den uppkom i Ryssland i början av 1900-talet. Teorin som beskrivs här utgår från Karlssons (1996) beskrivning och användning av aktivitetsteori. Aktivitetsteori har en omfattande teoribildning och det som beskrivs nedan är det urval av teorin som lämpar sig här.

Fem grundläggande termer inom teorin är: aktivitet, objekt, subjekt, mediator och kontext. En aktivitet definieras som en mänsklig process riktad mot ett objekt. Objektet beskriver det mål, problem, motiv etc som aktiviteten syftar till att påverka. Subjektet är den person som utför aktiviteten och mediatorn är det verktyg (abstrakt eller konkret) som aktiviteten utförs med. Kontext är den situation och omständighet som aktiviteten sker inom. För att förstå aktiviteten måste relationen mellan objekt, subjekt, mediator och kontext förstås. Relationen brukar visualiseras med hjälp av en triangel (figur 2.1).



Figur 2.1 Relationen mellan elementen i aktivitetsteori beskrivna med trianglar

När aktivitetsteori appliceras på PU²B-modellen är användaren subjektet och objektet är den uppgift som ska utföras. Funktionen är den mediator som användaren nyttjar för att utföra uppgiften och kontexten är den omgivning där användningen sker. Aktivitetsteori visar alltså att relationen mellan användaren, funktionen, uppgiften och omgivningen är det centrala som behöver förstås och beskrivas i utvecklingsarbetet. Teorin visar också tydligt att funktionerna finns för att användaren inte kan uppnå målen utan en mediator; dvs människan klarar inte av det på egen hand.

3 Modellen i stort

PU²B- modellen är uppbyggd i två dimensioner, dels en utifrån abstraktionsnivåer i ett system och dels en utifrån en uppsättning av modellobjektstyper, som var och en består av ett antal modellobjekt. Kapitlet beskriver hur detta bildar en helhet.

Modellarkitekturen (metamodellen) för PU²B-modellen har växt fram utifrån erfarenheten att befintliga modellspråk, ramverk och metoder har uppfattats som bristfälliga. Antingen genom att vara för vaga för designråd eller för omfattande för att snabbt kunna tillämpas. PU²B-modellen kan uppfattas som ett ramverk då den beskriver kopplingar mellan diagram. Den kan också uppfattas som en metod då den i vissa fall, som till exempel use case, föreslår en diagramtyp som ska användas för ett ändamål. Det är därför lämpligt att betrakta PU²B-modellen som en metamodell, en modell av modell, som kan och bör anpassas där den ska användas efter befintlig organisationsstruktur, produktutvecklingsprocess och typ av produkt. Med PU²B-modellen är det tänkt att systemingenjören eller HF-ingenjören snabbt ska kunna hänge sig åt just konceptutveckling och systemdesign, utan att fastna i komplicerat modellerande.

PU²B-modellen är användbar oavsett om du använder programvaruverktyg anpassade för MBSE eller enklare grafiska verktyg. Genom att använda PU²B-modellen ges förutsättningar att koppla affärsmål eller strategiska mål för ett system mot designen av systemet. Att förstå och kunna bryta ner dessa mål är en förutsättning för att effektivt tillämpa Product Life Cycle Management under systemdesign. Med hjälp av tydlig modellarkitektur kan det tidigt i designarbetet också tas hänsyn till olika systemaspekter som till exempel Human System Integration, System Safety och ILS (Gustafsson och Hedberg, 2013).

3.1 Abstraktionsnivåer i ett system

För att kunna bryta ner ett system på ett strukturerat sätt behövs det en indelning av abstraktionsnivåer. Detta är nödvändigt för att kunna föra diskussioner avseende behov, funktionalitet och lösningar på den abstraktionsnivån av ett system där diskussioner ger mest värde. Som traditionell tekniker är det lätt att snabbt börja diskutera en lösning i detalj, kanske redan innan övriga funktioner är utpekade eller alla behov identifierade. Vad som då händer är att det holistiska perspektivet riskerar att missas. Det finns en fara att den detaljerade lösningen får styra hur andra delar av systemet måste realiseras som konsekvens av löningen på det första problemet, i stället för att problemen värderas mot varandra och sedan välja en lösning som passar helheten. Desto senare i utvecklingsprocessen en design behöver göras om, desto högre blir totalkostnaden. Genom att strukturerat gå igenom abstraktionsnivåerna blir risken för omvälvande ändringar mindre. Det bör också påpekas att det är fullt möjligt att arbeta agilt även mellan abstraktionsnivåer.

Nedan presenteras de abstraktionsnivåer som modellarkitekturen i PU²B-modellen hanterar och de kan mycket väl ses som en "top down" systemdesign i tre processteg. Uppdelningen bygger på att fokus i utvecklingsarbetet skiftar, samtidigt som utvecklingsarbetet blir mer detaljerat ju längre processen framskrider.

3.1.1 Systemdefinition

Den första abstraktionsnivån visar, från systemets perspektiv, hur ömsesidig påverkan mellan systemet och dess omgivning ser ut. Syftet är att visa hur systemets omgivning påverkar och styr systemet. Målet är att identifiera intressenter och dess förväntningar på systemet. Genom att identifiera relationer på systemgränsen och intressenter (kan vara andra system) blir systemgränssnitten tydliggjorda. Systemdefinitionen klargör vad systemet ska leverera till slutanvändaren och i vilka sammanhang det förväntas att ske. På så sätt möjliggörs en koppling mellan önskad strategi, *kundvärde* lämpligen baserad på marknad och möjligheter, mot det som systemet bedöms kunna leverera, *systemuppgift* lämpligen baserat på företagets förmåga. I abstraktionsnivån arbetar man typiskt med systemet från ett ”Black-box” perspektiv, dvs att systemet betraktas som en helhet utan vetskap om hur dess inre struktur ser ut.

3.1.2 Produktdefinition

Arbetet i denna abstraktionsnivå syftar till att förstå den kontext som systemet ska kunna verka i och här utformas funktionell gränsyta mellan systemet och dess omgivning. Målet med produktdefinitionen är att tydligt visa vilka funktioner systemet ska ha och vilka krav som finns kopplade till dessa. Produktdefinitionen behandlar begrepp som anses vara självklara och vilka tillstånd som systemet förväntas kunna befinna sig i, betraktat utifrån. Slutmålet med Produktdefinitionen är att identifiera funktionalitet, samt att förklara hur systemet i stora drag ska leverera denna. Behov och övergripande ansatser för lösningar mognar fram utan att detaljerade realiseringar specificeras. Produktdefinitionen beskriver typiskt slutanvändarens upplevelse eller krav som till exempel tid och position men går inte in i lösningar eller realiseringsdetaljer av alternativ, som till exempel klocka, kompass och karta eller GPS. Görs detta fullt ut, möjliggörs en ”High level MIL” (Model In the Loop) där systemet kan simuleras och utvärderas på ett funktionellt plan. Det kan göras med hjälp av ett tillståndsdiagram som typiskt skulle kunna svara på frågeställningen: ”Om systemet hamnar i det här tillståndet aktiveras denna funktionalitet, vill vi ha det så?”.
På produktdefinitionsnivån definieras också systemets stora beståndsdelar och därmed bryts gränsen från en ”black box” perspektiv och går över i en ”grey box”. De övergripande systemdelarna och dess gränssnitt bör beskrivas, och kan till och med definieras beroende på var man vill dra gränsen mot nästa abstraktionsnivå. Det bör i denna abstraktionsnivå åtminstone övergripande beskrivas vilka delar i systemet som ska bidra till vilken funktionalitet och således görs en första inriktning för val och begränsningar avseende lösning av funktionalitet i systemet.

3.1.3 Arkitekturdefinition

I den tredje abstraktionsnivån är det första gången som systemet explicit närmar sig en realisering. Även om logisk arkitektur inte behöver representera de subsystem som faktiskt kommer att realiseras, är det här de första explicita designvalen för vilka gränssnitt som kommer finnas och hur flöden i systemet kommer att ske. Genom att ta fram arkitektur på hög nivå för både hårdvara och mjukvara, finns det en möjlighet att tidigt visa olika koncept som kan utvärderas mot varandra, för att optimera mot de inriktningar och avgränsningar som anses lämpliga för realisering. När beslut avseende önskad utvecklingskostnad, kritiska systemdelar och risker identifierats, har troligtvis inriktningen för valt koncept klarnat och det är i utvecklingsprocessen dags för nästa steg, detaljkonstruktion. Från det kompletta systemets perspektiv kan man typiskt börja betrakta den här abstraktionsnivån som ”white box” då dess interna delar och dess beroenden blir klargjorda.

3.1.4 Behövs abstraktionsnivåer?

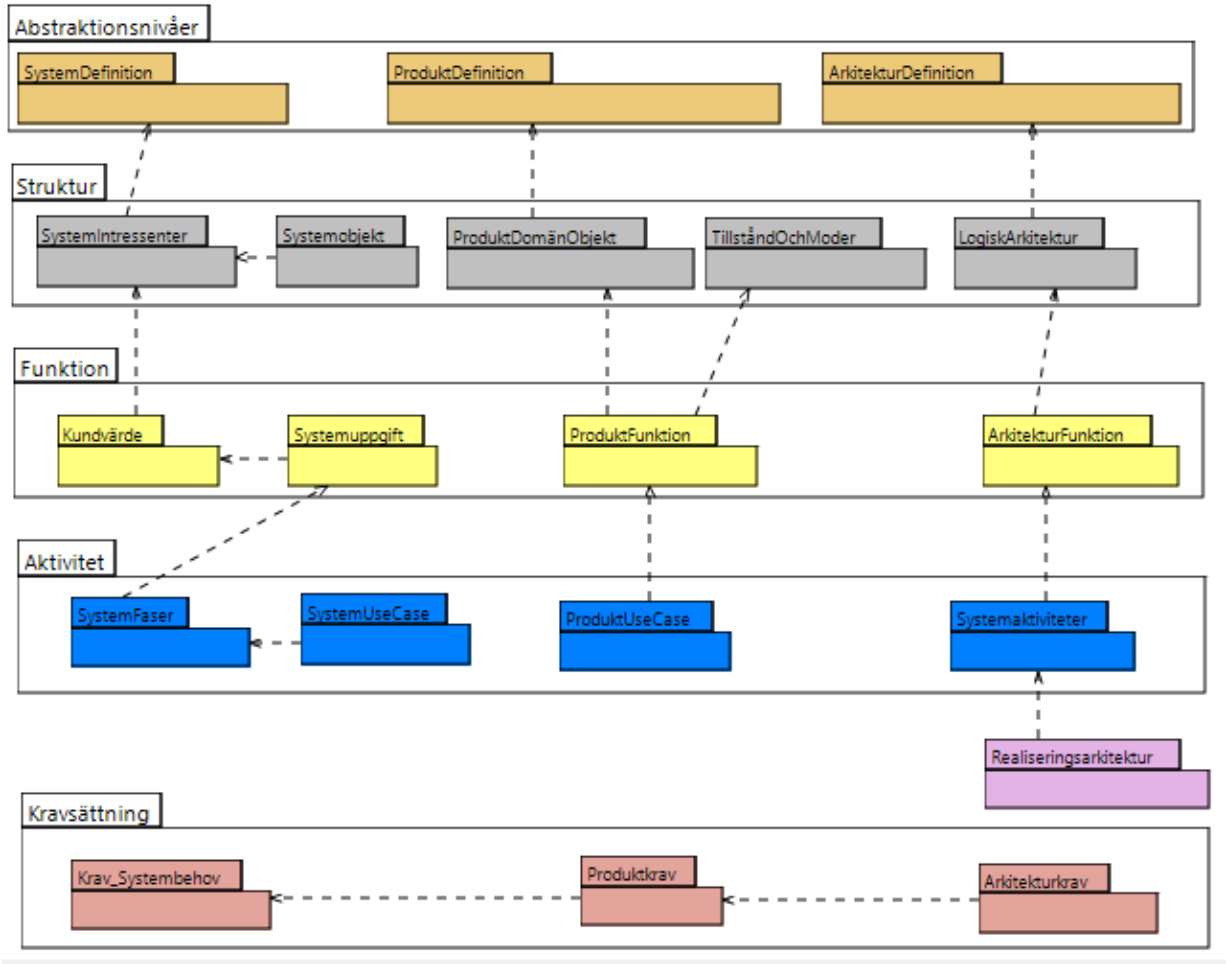
Systemdesign bör ske iterativt. Erfarenhet visar att det ofta finns frågeställningar av liknande karaktär som kan ha en koppling till varandra men som var för sig bör lösas på en passande abstraktionsnivå för att erhålla en optimal systemdesign. Det finns ett tydligt behov av att klargöra funktionalitet för systemet, enskilda delar av produkten och komponenter var för sig, även om de självklart också är kopplade till varandra. Till exempel kan en interaktionsfråga hanteras på abstraktionsnivån produktdefinition för att klargöra hur en användare förväntas interagera med produkten. Dessa frågor har då både en koppling till systemdefinition, utifrån hur systemet förväntas förhålla sig till omvärlden och upplevas samtidigt som det i sig ställer krav på arkitekturdefinition och inte minst på realiseringsarkitektur. Om det till exempel eftersträvas ett intuitivt användargränssnitt för touchskärm, eller låt oss säga en förberedelse för hologram, då måste hela systemets infrastruktur klara av detta. Ser man inte tydligt kopplingar mellan önskemål, intention och realisering blir det svårt att optimera systemdesignen.
Ett av de viktigaste bidragen med PU²B-modellen är att den tydligt visar att det finns liknande frågeställningar på olika abstraktionsnivåer som bör besvaras, för att ett system ska kunna utvecklas effektivt. Om detta behov inte erkänns eller redogörs för, är risken att produktutveckling bedrivs oekonomiskt. Problem riskerar att adresseras till fel abstraktionsnivå, vilket i sin tur kan bidra till ”mörkläggning” då problem inte lyfts fram. Vidare riskerar problem att ”falla mellan stolar” om det inte i utvecklingsarbetet går att klargöra vilken nivå problem bör lösas på och tydligt peka ut en ”ägare” till dem. Essensen av PU²B-modellen är att just att tydliggöra abstraktionsnivån

produktdefinition. Det är en abstraktionsnivå som traditionellt är underskattad men som kommer behövas i allt större utsträckning, inte minst på grund av produkters ökade komplexitet. I ”gammaldags” produktutveckling har det varit vanligare att krav på produkter men också lösningar föreslås direkt av säljare eller av ”chefsingenjörer” på hierarkisk toppnivå. Det är möjligt att göra så länge produkten är relativt okomplicerad, stabil och arbetsgruppen för utvecklingen är liten. När någon av dessa förutsättningar inte längre är uppfyllda behövs det en struktur som kan hantera flera abstraktionsnivåer på ett explicit sätt. Annars ökar risken för missförstånd. I de fall där man börjar i en ände av det komplexa problemet och ”gör något” för att sedan bygga vidare på den tekniska ansatsen, riskerar man inte bara att tappa helhetssynen utan också förmåga till systemoptimering. Om företaget väl lyckas få ut en produkt som uppfyller kraven, är den typiskt svår att anpassa till nya krav och bygga vidare på i nästa strategiskt planerade utvecklingssteg.

Att produktdefinitionen betonas extra tydligt från Human Factors perspektiv beror på att de frågorna alltid har varit centrala för att kunna designa från användarperspektiv. Därmed anses det nu ha visats att denna traditionellt underskattade abstraktionsnivå är nödvändig framöver. Samtidigt blir det tydligt att det genom att tidigt ta höjd för Human Factors tidigt i utvecklingsprocessen är det samma som att arbeta enligt Lean Product Development.

3.2 Modellobjektstyper

För att tydligt peka ut vad som avses hanteras på respektive abstraktionsnivå har benämningen modellobjektstyp valts för att representera en typ av objekt som återkommer i olika abstraktionsnivåer. I figur 3.1 visas de tre modellobjektstyper som modellen använder: struktur, funktion och aktivitet placerade i de tre abstraktionsnivåerna.



Figur 3.1 Strukturen och objekten i PU²B- modellen

Det är viktigt att påpeka att PU²B-modellen i stora drag representerar en metamodell, det vill säga en modell av en modell. För de system som modelleras är det viktigt att alla objekt som används anpassas efter det specifika systemets behov. Det är också viktigt att påpeka att PU²B-modellen är tänkt som ett stöd till en produktutvecklingsprocess och därför lämpligen bör anpassas efter den. Till exempel finns det ofta i utvecklingsprocesser redogjort för när olika gränssnitt för ett system bör tas fram och hur de bör beskrivas. De kan då införas som ytterligare objekt i PU²B-modellen och bör, beroende på typ av gränssnitt, placeras i den mest lämpliga abstraktionsnivån. Externa gränssnitt skulle kunna läggas som objekt i abstraktionsnivån Systemdefinition, medan de övergripande interna gränssnitten skulle kunna redogöras för i Produktdefinition.

Nedan presenteras varje modellobjektstyp, först med en övergripande beskrivning och sedan med en presentation per objekt (figur 3.1). En mer detaljerad presentation av kommer i kommande kapitel.

3.2.1 Strukturobjekt

Strukturobjekt används för att beskriva de objekt som bidrar till ordning och reda men som i sig själva inte representerar funktionalitet eller aktivitet. Det kan vara intressenter som använder systemet, faser som systemet förväntas verka i eller olika tillstånd för systemet. De strukturobjekt som används i PU²B-modellen är:

- Systemintressenter – vilka visar systemets intressenter.
- Systemobjekt – som presenterar systemet och dess omgivning.
- Produktdomänobjekt – redogör för de beståndsdelar som ett system antas bestå av. Det är typiskt högnivådelar eller funktionella delar av systemet (Det är dessa delar som kan utgöra simuleringsentiteter för simulering på funktionell nivå.)
- Tillstånd och moder – redogör för de tillstånd som systemet ska kunna befinna sig i.
- Logisk arkitektur – visar hur delar av systemet behöver kopplas ihop för att kunna realisera de önskade funktionerna identifierade i tidigare abstraktionsnivåer.

3.2.2 Funktionsobjekt

Funktionsobjekt används för att beskriva funktionalitet och vad som förväntas kunna levereras av systemet. Objekten är tydligt inriktade på att just visa den funktionalitet som systemet och dess delar ska kunna leverera. Funktionsobjekt som används i PU²B-modellen är:

- Kundvärde – vilken redogör för systems förmåga i förhållande till sin omgivning.
- Systemuppgift – som visar vad för aktiviteter som systemet förväntas kunna göra för att kunna leverera önskad förmåga.
- Produktfunktion – presenterar de funktioner som en produkt behöver för att kunna utföra en uppgift i en given kontext.
- Arkitekturfunktion – redogör för de funktioner som den kommande lösningen behöver för att kunna realisera produktens förväntade funktionalitet under givna förutsättningar. Den kan också betraktas som en delsystemarbitrering där det framgår vad som ägs av den mekaniska arkitekturen och vad ägs av den elektriska arkitekturen.

3.2.3 Aktivitetsobjekt

Aktivitetsobjekt används för att tydliggöra de aktiviteter som systemet förväntas utföra på de olika nivåerna. Inom denna modellobjektstyp är det särskilt tydligt att samma typ av frågeställning hanteras på alla abstraktionsnivåerna. De aktivitetsobjekt som används i PU²B-modellen är:

- System-use case – redogör för aktivitet genom att tydliggöra den ”yttre” kontexten som systemet ska kunna verka
- Product-use case – redogör för aktiviteter genom att tydligt beskriva kontexten som själva produkten ska kunna verka i
- Systemaktiviteter – är de aktiviteter som systemet förväntas utföra baserad på dess logiska arkitektur

3.3 Systemvyer

Att använda PU²B-modellen med abstraktionsnivåerna är tänkt som ett stöd för en kontrollerad utveckling och för att enkelt kunna introducera Model Based Systems Engineering (MBSE). En av fördelarna med att använda en struktur som PU²B-modellen, är just att tydliggörandet av kopplingar mellan olika delar (objekt) i ett system. För att kunna visa dessa kopplingar används här begreppet systemvyer. En systemvy är en mappning där två olika modellobjekt ritas i samma grafiska representation och där dess relationer tydliggörs. Som inriktning är det en rekommendation att inte koppla fler än två modellobjekt i samma systemvy då det riskerar att blir mer svårtolkat. En systemvy bör tas fram för ett specifikt syfte. Ett exempel skulle kunna vara att under arbete i abstraktionsnivån Systemdefinition koppla identifierade Systemfaser till de System-use case som valts. Detta skulle i sin tur underlätta identifiering av Produktfunktion och Produkt-use case under arbete i abstraktionsnivån Produktdefinition. Vika systemvyer som anses lämpliga beror på det enskilda systemet.

I samband med systemvyer är det lämpligt att varna för ”modelldjävulen”. Det går att göra en modell flera gånger större än vad som är lämpligt för utvecklingsarbetet. Bara för att det går att modellera samband är det inte lämpligt att modellera alla samband överallt. Vissa samband kanske kan finnas som krav på olika nivåer istället för att visas med kompletta vyer av samband eller objekt.

4 Systemdefinition

Den första abstraktionsnivån används för att, från systemets perspektiv, visa på verkan mellan systemet och dess omgivning. Syftet är att beskriva hur systemets omgivning påverkar och styr systemet och tvärtom, vad systemet förväntas leverera ut till omgivningen. Målet är att identifiera intressenter och dess förväntningar på systemet, samt hur systemet verkar mot sin omgivning. Genom att identifiera relationer på systemranden och intressenter (som även kan vara andra system) blir systemgränserna tydliggjorda. För att kunna arbeta i denna abstraktionsnivå behövs nedan beskrivna modellobjekt.

- Systemintressenter
- Systemobjekt
- Kundvärde
- Systemuppgift
- Systemfaser
- System-use case
- Krav systembehov

4.1 Systemintressenter

Systemintressenter är de som har intresse av systemet på ett eller annat sätt. Det innefattar exempelvis både ägare av företag eller elektroniska tjänster, som inte behöver ha någon egentlig kontakt med systemet, likväl som slutanvändare. Med slutanvändare menas även personer som kan drabbas i andra hand av att systemet används som till exempel en IT-tekniker (jämför också ”passiva rökare”). Systemintressenter kan även vara ett annat system som förväntar sig något av systemet som är under utveckling. Exempelvis kan en elektronisk karta kan använda data från en GPS, givet att den elektroniska kartan och GPS betraktas som skilda system. Desto tidigare intressenterna identifieras desto enklare blir det att resonera kring vad som förväntas av systemet och när det förväntas. Därmed blir det enklare att gå in i nästa abstraktionsnivå för att definiera vad som faktiskt avses göras med systemet. Det blir också enklare att prioritera kärnan av systemet och vad det ska leverera. Modellobjekt för Systemintressenter kan skapas som ett klassdiagram (UML/SysML), objekt i form av boxar eller på enklast sätt som en lista eller del av tabell. Det är lämpligt att även med några ord kort redogöra för den kontext där respektive intressent förväntas finnas.

4.2 Systemobjekt

Dessa objekt används för att ge en första bild av den värld som systemet som helhet förväntas verka i. Målet är tydliggöra systemet gentemot den omgivning som det förväntas interagera med. Till exempel kan ett värmesystem för ett hus förväntas vara installerat i ett hus. Värmesystemet kanske hämtar eller lämnar medium från sin omgivning eller behöver kraft för att fungera. Alla dessa delar som systemet som helhet förväntas interagera med, vare sig det är enkelriktad påverkan eller dubbelriktad interaktion, kan beskrivas som systemobjekt. Arbete med detta objekt är ett viktigt steg mott att identifiera ett systems gränssnitt mot omvärlden.

4.3 Kundvärde

Kundvärde ska beskriva det som systemet ska leverera till sin omgivning. Kundvärde är kärnan för ett systemets existens och motsvarar det som kunderna förväntar sig att få ut av systemet Köper man en produkt för att man vill få ut en praktisk nytta eller är det i någon form en statussymbol eller både och? Att tydligt peka ut och definiera kundvärde är det första steget mot att internt inom den egna organisationen ensa synen på det som man avser att leverera. Desto mer entydigt det specifika systemets förmågor beskrivs, desto enklare blir det fortsatta arbetet med produkten, både avseende systemdesign och detaljkonstruktion**.** Dessa modellobjekt skapas lämpligen som klassdiagram (UML/SysML) eller på enklast sätt som objekt i form av boxar och kan delas upp i en hierarkisk nivå.

4.4 Systemuppgift

Arbetet med modellobjektet Systemuppgift syftar till att presentera det som systemet ska utföra. Om modellobjektet Kundvärde är det som avses säljas in, är modellobjektet Systemuppgift det som systemet konkret ska göra för att kunna uppfylla önskad förmåga. Ett perspektiv på detta kan vara att modellobjektet Systemuppgift definierar modellobjektet Kundvärde. Modellobjekt Systemuppgift skapas lämpligen som klassdiagram (UML/SysML) eller på enklast sätt som objekt i form av boxar och kan delas upp i en hierarkisk nivå. Ett sätt att arbeta fram Systemuppgifter är att skapa en systemvy genom att koppla den lägsta hierarkiska nivån av modellobjekten Kundvärde (givet att flera nivåer finns) mot den översta nivån av Systemuppgift.

4.5 Systemfaser

Systemfaser används på den första abstraktionsnivån för att underlätta arbetet med att följa det tänka systemet genom hela dess produktlivscykel. Traditionellt, även inom ämnesområdet Human Factors, är det lätt att fokusera på slutanvändaren och dess tänkta interaktion. Genom att tydligt visa det tänka systemets hela livscykel, blir det enklare att identifiera användningsfall som kan ge vidare guidning för konceptutveckling och design av produkten. Det skulle till exempel kunna vara så att om man designar den kommande produkten på ett speciellt sätt eller redan i produktion förser produkten med ett extra skydd, så kanske man underlättar hanteringen av transport från fabrik till återförsäljare. Denna typ av krav och förutsättningar blir enklare att identifiera och arbeta med om man tidigt kartlägger och skriver ner sin förförståelse av det kommande systemet.

4.6 System-use case

System-use case används för att tydliggöra den första ansatsen av användarkontext. Första steget med att identifiera dessa use case är lämpligen att skapa use case diagram. För den som inte är invigd i UML/SysML kan man beskriva use case diagram som en struktur av bubblor som visar vad det ska gå att göra med systemet. Use case skrivs till skillnad mot tidigare System task från användarens perspektiv. Varje use case döps som en aktivitet i presens och bör kopplas eller länkas samman med andra use case så att sammanhang för systemet blir tydligt. Både tänkta flöden och relationer mellan use case, som till exempel specialfall, kan visas i use case diagram. Det kan inledningsvis vara svårt att få en klar och tydlig stringens i denna typ av modellering men ett riktmärke är att varje överliggande use case i en hierarkisk struktur inte har fler än 7+-2 use case kopplade till sig i underliggande nivå. Denna struktur av objekt kräver oftast flera bearbetningar och omstruktureringar. Ytterligare ett råd att tänka på kan vara att inte initialt försöka fånga precis alla tänkbara aspekter av systemet utan just de perspektiv som avses vara mest relevant för en lyckad och framgångsrik produkt.

Om det finns ytterligare behov av att klargöra vad ett use case betyder, kan man förtydliga dessa genom aktivitetsdagram. Det är lämpligt att göra detta för de use case som ligger längst ner i den hierarkiska trädstrukturen (de så kallade ”löven”). Det som görs rent praktiskt i ett aktivitetsdiagram är att man skriver ner de aktiviteter som man förväntar sig finnas inom respektive use case och sedan länkar dessa aktiviteter till varandra.

Att beskriva vilka aktiviteter som förväntas finnas i ett use case, är en bra start med arbetet att tydligt strukturera och definiera use case. Det finns ofta en uppenbar koppling mellan aktiviteter och den tänkta funktionella designen och/eller realiseringen av denna. Aktiviteter kan återkomma som beståndsdelar inom olika use case och det är dessa delar som blir betydande för att lyckas hålla Human System Integration arbetet konsekvent genom hela designen.

4.7 Krav systembehov

Krav på systemnivå kan utformas som önskemål på systemet eller vad det övergripande förväntas kunna leverera.

4.8 Exempel stekspade: Systemdefinition

För att kunna presentera modellen på ett konkret sätt har vi valt att exemplifiera arbetet i de olika abstraktionsnivåerna genom att modellera en stekspade. För varje abstraktionsnivå visas därför de modellobjekt som anses behövas med några exempel. Observera att dessa exempel endast är avsedda för att guida och för att inte ”svämma över” har exemplen beskurits och innehåller inte alla detaljer.
Det är också viktigt att komma ihåg att resultatet av en modellering kan se väldigt olika ut beroende på vem som modellerar. Ofta finns inom en organisation kan finnas ett antal olika grundantaganden om vad ett system eller en produkt ”egentligen” är till för vilket påverkar modelleringen. Det finns heller inga ”rätt eller fel” sätt att modellera, bara mer eller mindre lämpligt beroende på vad man är ute efter att visa.

4.8.1 Stekspade: systemintressenter

För att kunna skapa en modell är det viktigt att veta vilka som har intresse av modellen. I detta exempel listas i all enkelhet de övergripande intressenter som beaktas för fortsatt modellering.

- Tillverkare (Original Equipment Manufacturer, OEM)
- Återförsäljare
- Slutanvändare
- Passiv användare (den som äter maten eller ser spaden)

Det är viktigt att påpeka att desto fler intressenter som identifieras underlättar det kommanden arbete med att identifiera use case. Ett sätt att utveckla struktur kan vara att sätta upp olika roller och deras inbördes kopplingar. Ägare, verkställande chef och inköpare är exempel på roller som skulle kunna finnas inom intressenten tillverkare och som alla har olika perspektiv på vad som är viktigt för produkten. Desto tydligare de mest väsentliga sambanden visas, desto mer hänsyn kan tas i form av medvetna beslut.

4.8.2 Stekspade: systemobjekt

Nedan ansatser ger några exempel på denna typ av modellobjekt. De bör klargöra de inriktningsbeslut som tas tidigt, lyckas dessa bli tydliga ges en bra grund för att ta fortsatta beslut i utvecklingsprocessen. Vår stekspade kommer på ett eller annat sätt att anta en fysisk form som vi nu väljer att kalla stekspade. Den kommer efter tillverkning att förpackas, varpå den kommer få en relation till förpackningen. Om vi redan nu har för avsikt att göra en exklusiv stekspade bör den till exempel få en egen unik förpackning för varje exemplar. Till exempel kan förpackning potentiellt underlätta både transport och utställning hos återförsäljare om den är rätt konstruerad, med lämplig form eller med genomskinlig sida. Stekspaden kommer behöva komma i kontakt med både mat och andra matlagningsverktyg av olika material och att tydlig skaffa sig en första bild av dessa underlättar kravställning. Den kommer att behöva rengöras och ett antagande är att det kommer ske med vatten, men ska den tåla diskmaskin?

Nedan listas de första systemobjekten som ska vara vägledande:

- Stekspade
- Förpackning
- Användare
- Mat
- Matlagningsverktyg i dess förväntade kontext
- Rengöringsmedel / vatten

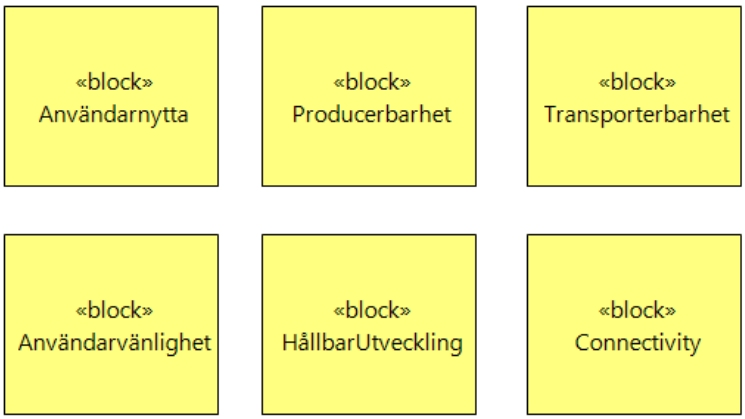
4.8.3 Stekspade: kundvärde

För stekspaden väljs två övergripande kundvärde i form av köksidentitet och användarnytta (figur 4.1). Utformas stekspaden rätt, är den tänkt som en artefakt som positivt bejakar den kommande kockens köksidentitet. Har den rätt användarnytta (utility) blir den miljövänlig, lätt att producera, lät att transportera och visa upp i handeln samt lätt att använda för slutkunden.

Figur 4.1 Kundvärde för stekspaden

4.8.4 Stekspade: systemuppgift

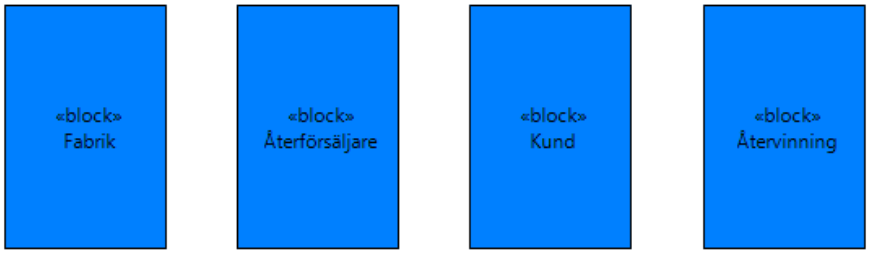
För att stekspaden ska kunna leverera önskade förmågor har följande systemuppgifter identifierats (figur 4.2). Dessa systemuppgifter visar tydligt de grundantaganden som finns i organisationen och återspeglar ofta företagets strategi så väl som de grundläggande värderingarna. Om det tänkta systemet klarar av dessa uppgifter så tror man från ”top down”-filosofin att produkten blir framgångsrik.



Figur 4.2 Systemuppgift för stekspaden

4.8.5 Stekspade: systemfaser

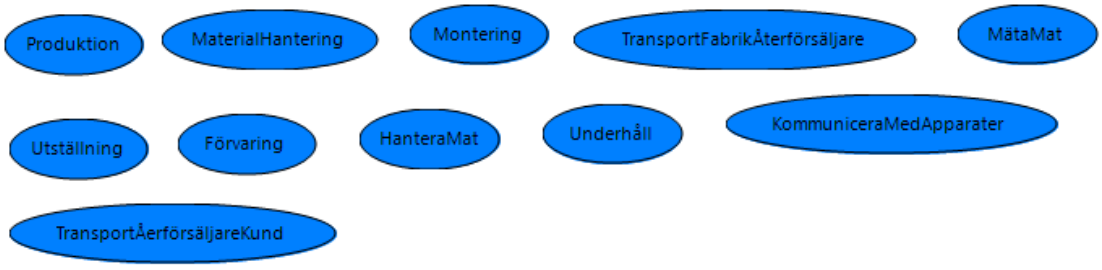
Dessa faser visar övergripande produktens livscykel (figur 4.3). För stekspaden har vi valt att hålla detta enkelt genom att bara ta hänsyn till de översta nivåerna. Självklart kan dessa utvecklas mer om det, till exempel finns intresse gå djupare in i aspekten av tillverkning. Kanske det till och med är intressant att genom modellering utreda hur produktion av stekspaden skulle kunna ske optimalt.



Figur 4.3 Systemfaser för stekspaden

4.8.6 Stekspade: system-use case

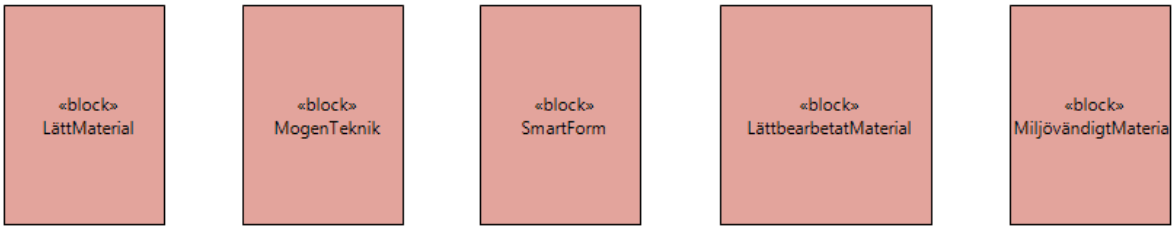
System-use case beskriver det sammanhang som det framöver kommer att behövas tas hänsyn till under systemdesign (figur 4.4). Det som fångas upp som systemets övergripande aktiviteter för att kunna utföra sin uppgift och leverera förmåga.



Figur 4.4 System-use case för stekspaden

4.8.7 Stekspade: krav systemkrav

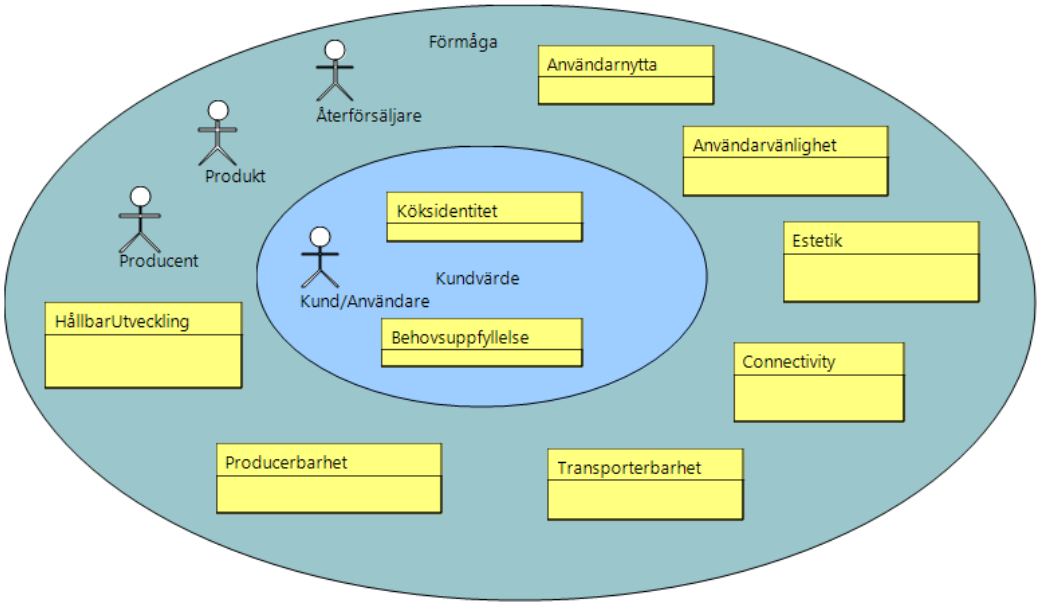
För att systemet ska kunna designas efter önskemål, listas här initialt det som kan betraktas som önskemål på systemet (figur 4.5). I vilken utsträckning de går att uppfylla bör lämpligen följas upp på kravställning i kommande abstraktionsnivåer.



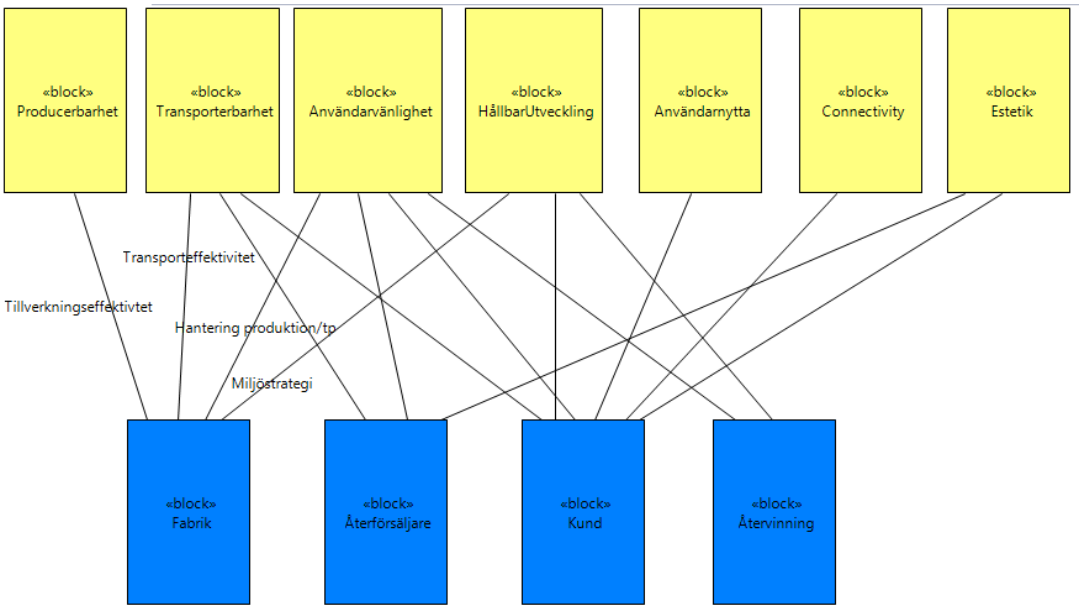
Figur 4.5 Systemkrav för stekspaden

4.8.8 Stekspade: systemvyer systemdefinition

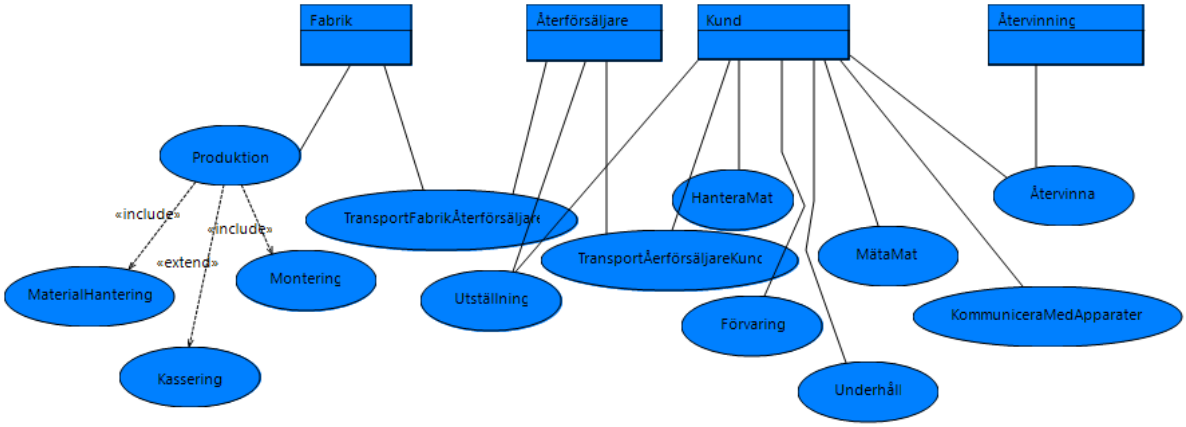
Nedan presenteras de systemvyer som valts inom abstraktionsnivån Systemdefinition (figur 4.6-19). Systemvyer kan visas på olika sätt och målet med systemvyer är att just koppla samman de modellobjekt som anses tillföra mer information, för klargörande av ställningstagande. Först visas kopplingen mellan intressenter och systemuppgifter med hjälp av en representation av ekosystem. De andra vyerna visas genom så kallad mappning som grafiskt visar kopplingar mellan modellobjekt.



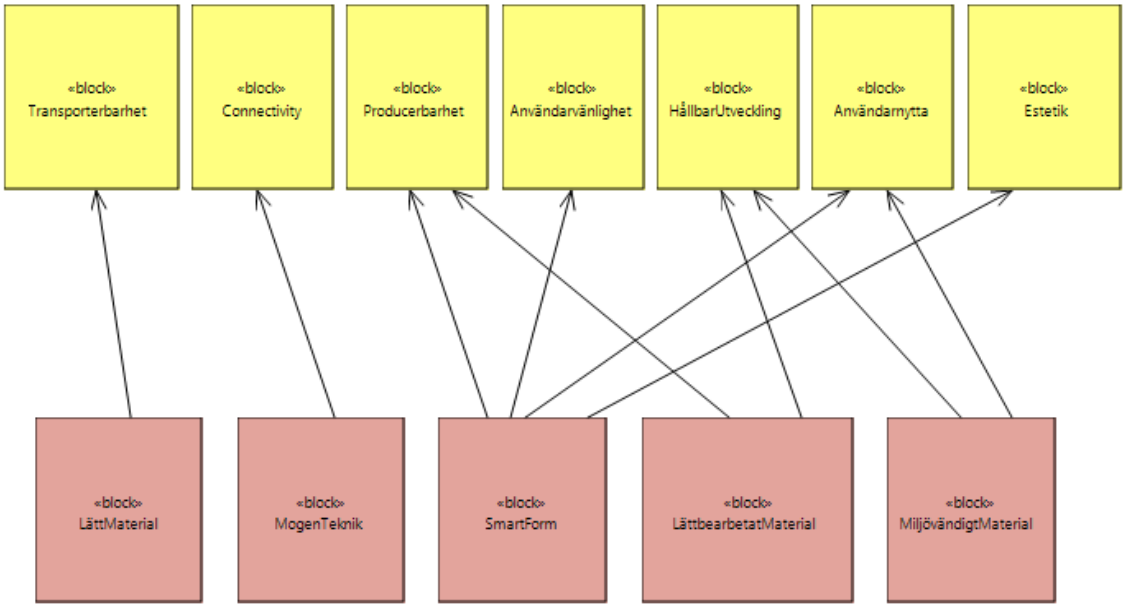
Figur 4.6 Koppling kundvärde mot systemuppgift (förmåga)



Figur 4.7 Koppling systemuppgift mot systemfaser



Figur 4.8 Koppling systemfaser mot system-use case



Figur 4.9 Koppling systemuppgift (förmåga) mot systemkrav

5 Produktdefinition

Arbetet i den andra abstraktionsnivån syftar till att förstå den kontext som systemet ska kunna verka i. Benämningen produkt har valts för att förtydliga att fokus ligger på det som faktiskt ska kunna säljas och ett nyckelbegrepp för denna abstraktionsnivå är funktionalitet, dels funktionalitet för den tänkta produkten och dels funktionella gränssnitt. Målet med produktdefinitionen är att tydligt visa vilka funktioner systemet ska ha och att koppla krav till dessa. Utöver funktionalitet behandlar produktdefinitionen begrepp som vilka tillstånd som systemet förväntas kunna befinna sig i. För att kunna arbeta i denna abstraktionsnivå behövs nedan beskrivna följande modellobjekt:

- Produktdomänobjekt
- Tillstånd och moder
- Produktfunktion
- Product-use case
- Produktkrav

5.1 Produktdomänobjekt

Målet med dessa modellobjekt är att beskriva de delar som initialt kan anses självklara men som likväl är väsentliga delar av produkten och dess funktionalitet. För ett hus skulle exempel på denna typ av objekt kunna vara fönster, väggar eller tak. Det är svårt att tänka sig ett hus utan dem, samtidigt som det på denna nivå av abstraktion fortfarande finns mycket funktionalitet som sikt igenom eller isolering som inte är definierad. Å ena sidan kan det anses att produkten inte längre är lösningsoberoende och samtidigt finns det mycket kvar att definiera både avseende funktionalitet och design. Två-glas eller tre-glas fönster, tegel eller trävägg är exempel på design som bör utformas efter att funktionalitet fastställs för att lösa de behov som identifierats rent funktionellt. Produktdomänobjekt bör behandla den typen av objekt som tas för givet i systemet samtidigt som de kan betraktas som ett första steg mot en lösning. Någonstans måste man börja och genom att medvetandegöra begreppet ”lösningsoberoende” kan det, genom arbete med dessa objekt, bli väldigt tydligt vad som ”bor i väggarna” och vilka lösningar som därmed är att betrakta som självklara. Även om man borde förhålla sig så oberoende en lösning som möjligt i syfte att lämna fler möjligheter för arkitektur och design, så är det inte praktiskt möjligt i många fall. Väljs dessa objekt med omsorg, kan de mycket väl bli de första byggstenarna för att kunna utföra funktionell simulering. Vidare är arbetet med produktdomänobjekten centralt för att identifiera systemets interna gränssnitt.

5.2 Tillstånd och moder

5.2.1 Tillstånd

Modellobjektet Tillstånd beskriver vilka olika tillstånd ett system kan befinna sig i. Det är viktigt att först betrakta hela systemet utifrån sin yttersta systemrand. Att definiera tillstånd är därför ett bra sätt att kunna identifiera gränssnitt, både mellan systemet och dess omgivning och mellan delar i systemet. Ofta har ett system en eller flera grundtillstånd och det är övergångar mellan dessa som blir kritiska för systemet. Modellobjektet Tillstånd syftar dock främst till att definiera just själva tillstånden, medan övergångar mellan dem lämpligen beskrivs som specialfall av use case (se senare beskrivning). Beroende på hur många olika tillstånd som finns kan även denna objektklass delas upp i hierarkiska träd. Om det dock visar sig att det är enskilda komponenter eller delar av enskilda funktionella objekt som står för skillnaden i ett tillstånd, kan det vara lämpligt att i stället använda begreppet mod (Se nästa avsnitt).

5.2.2 Mod

Mod är i PU²B-modellen att betrakta som ett specialfall av ett tillstånd. Målet är att tydligt visa behovet av att uppnå ett tillstånd genom att bara ändra på en liten del av systemet. Att identifiera moder görs typiskt en bit in i designfasen då funktionella objekt eller till och med komponenter definieras. I vissa fal finns det dock tydligt utpekade delsystem, som kanske till och med är att betrakta som COTS (Commercial Off The Shelf) och har dessa redan definierade delsystemtillstånd kallas de i PU²B-modellen för moder.

5.3 Produktfunktion

För att kunna göra en första ansats för detaljerad design är det lämpligt att identifiera funktionella objekt. Dessa objekt är svåra att hantera, speciellt för den ”traditionelle” teknikern som gärna vill lösa ett problem direkt med en känd artefakt i form av en komponent eller mjukvara. Målet med funktionella objekt är att beskriva vad som anses behövas för att systemet ska kunna verka enligt de tidigare definierade modellobjekten. Genom att beskriva dessa just som funktionella objekt och inte direkt som realiseringar, öppnas vägen för alternativa lösningar. Det är ofta genom att identifiera behov, vilka tas för givna eller som bara är föreställningar avseende systemet, som förutsättningar för systemet blir tydligt. Ett konkret exempel kan vara funktionen att hålla reda på ett systems tid och plats. Ett sätt att realisera detta är att använda en GPS och ett annat är att använda traditionell papperskarta, kompass och ett armbandsur. I grunden kan båda dessa sätt vara lika bra för att klara av funktionen men vilket som är att betrakta som bäst beror på vilken systemuppgift som ska lösas och på vilket sätt (system use case) det förväntas ske för att tillgodose kunden (systemuppgift).

Finns det utpekade COTS som avses utgöra en del av systemlösningen, bör dessas funktioner abstraheras fram för att designen av systemet ska bli optimal.

Identifierandet av funktionella objekt är en bra förutsättning för att kunna hantera en produkt som det avses arbetas med under lång tid. Att resonera kring systemet i form av funktionella objekt ger en bra förutsättning för att förstå nya förutsättningar som kan uppstå genom vidareutveckling av enskilda komponenter. Det är också ett bra första steg för att identifiera förbättringsförslag och vidareutvecklingar av produkten.

Ytterligare en fördel av att ha den funktionella strukturen tydlig är förmågan att hantera strategiska beslut. Ibland kan det inses att en funktion är eftersträvansvärd men att den kanske inte självklart ska utvecklas inom den egna produkten. Att ha timerfunktion till en kaffekokare kan vara önskvärt men är kanske inget som ligger inom avgränsningen för den egna produkten. Exemplet visar tydligt hur strategiska inriktningar påverkar systemdesign. Ligger inte timer inom produktdefinitionen kanske kaffemaskinstillverkaren kan alliera sig med en timertillverkare och erbjuda en specialtillverkad timer som merförsäljning. Vad som avgör borde lämpligen vara företagets egen färdighet och kompetens inom vald realiseringsdomän. Är inte kaffemaskinstillverkaren duktig på styr- och reglerprocesser för timer vill de sannolikt alliera sig med annat bolag. Kan de enkelt hantera tekniken utan större merkostnad bygger de sannolikt in funktionaliteten själva. Frågan om vad som ska betraktas som bolagets kärnprodukt blir väldigt viktig. Desto längre bort från kärnvärdet men större potentiell vinst desto större är intresset för att hitta en partner. Desto närmre kärnverksamhet och bedömd stor vinst desto större är chansen att den nya funktionaliteten integreras.

5.4 Produkt-use case

Med hjälp av modellobjektet Produkt-use case beskrivs de aktiviteter som behöver ske med eller inom produkten. Typiskt behöver övergångar mellan olika tillstånd beskrivas. Vidare är det lämpligt att med Produkt-use case förtydliga Produktfunktion.

Att definiera Produkt-use case är en väsentlig del av arbetet med Human System Integration. Genom PU²B-modellen erbjuds en tydlig koppling från Systemuppgift till System-use case som uttrycker behov för identifiering av Produktfunktion som i sin tur pekar ut Produkt-use case. Att ha tydliga Produkt-use case är det första steget mot att kunna möta de förväntningar som finns från användare och de är också en viktig förutsättning för att kunna återspegla en produkts värden. Det ökar möjligheterna att beslut kan fattas medvetet och explicit samt det är enklare att hålla en konsekvent linje från identifiering av funktionalitet till design. Även om det finns en uppenbar koppling mellan use case och HSI (Human System Integration) kan Produkt-use case skrivas från ett rent tekniskt perspektiv.

När Produkt-use case definieras är det viktigt att fånga alla de områdena som berör en produkt genom hela dess produktlivscykel. Hela produktens väg från montage till vanligt bruk eller speciellt bruk, samt service och skrotning bör beaktas.

Vidare bör det nämnas att en tydlig funktionell struktur underlättar kostnadseffektivt och transparent kvalité- och systemsäkerhetsarbete. Har man genom en tydlig struktur en första ansats för övergripande Tillstånd och önskad Produktfunktionalitet kan man tydligt koppla dessa till Produkt-use case och arkitektur. Därmed är det enkelt att förstå konsekvenser av bortfall för funktionalitet av enskild arkitekturell del och dess realisering. Detta är för exempel vid arbete med Failure Mode Effect Analysis (FMEA) väldigt centralt.

5.5 Produktkrav

På denna nivå kravställs vanligen funktionalitet och de förmågor/ egenskaper som systemet ska leverera.

5.6 Exempel stekspade: Produktdefinition

Nedan fortsätter exemplet med stekspaden utifrån de modellobjekt som presenterats för produktdefinition.

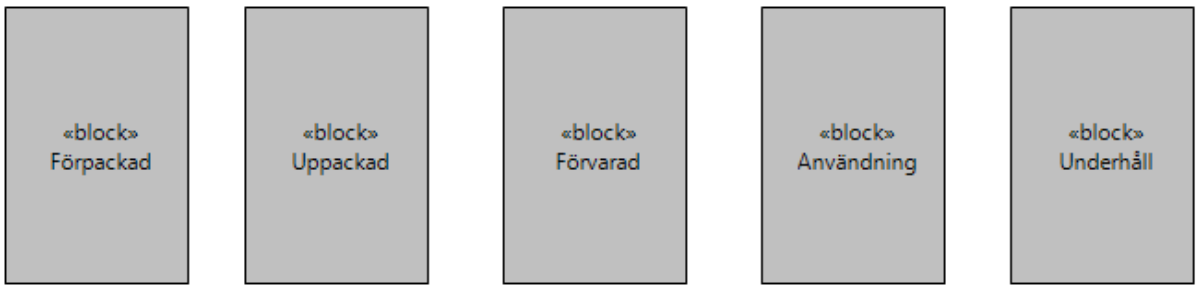
5.6.1 Stekspade: produktdomänmodell

För stekspaden var det viktigt att klargöra att den fysiskt liknade en spade (figur 5.1). Den kunde varit utformat på olika sätt, kanske till och med som en stav eller bara som en platta. Men att utgå ifrån en spadform med en handtagsdel och en arbetsdel valdes tidigt som inriktning.

Figur 5.1 Produktdomänmodell för stekspaden

5.6.2 Stekspade: tillstånd och moder

Det eftersträvades att tydligt visa de olika tillstånden som den tänkta stekspaden förmodades kunna anta. I detta exempel visas tillstånden fram till den tänkta användningen i ett hem men skulle givetvis för bästa helhetsperspektiv beaktas genom hela sin livscykel från färdig produkt till återvinning (figur 5.2).

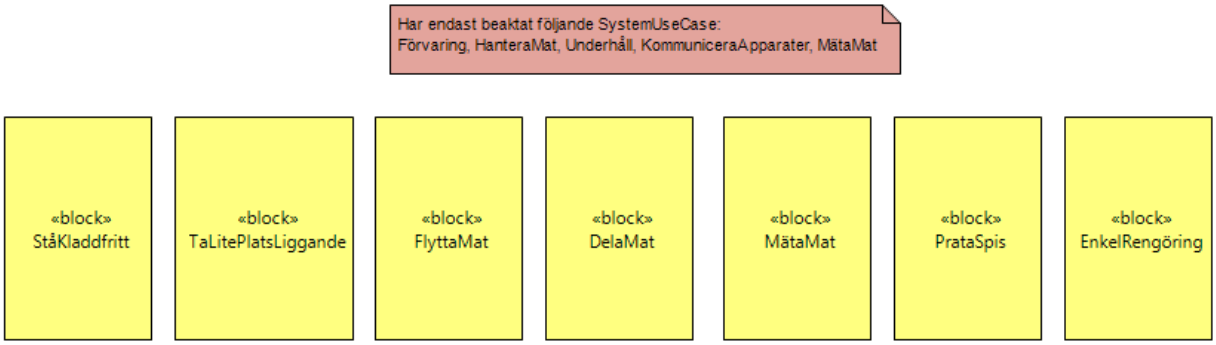


Figur 5.2 Tillstånd och moder för stekspaden

5.6.3 Stekspade: produktfunktion

Nedan presenteras de objekt som identifierats för stekspaden (figur 5.3). Det är viktigt att poängtera att dessa exempel inte är kompletta för hela stekspaden, utan är tänka som exempel för att underlätta förståelsen för modellen och dess struktur. I nedan exempel har det gjorts en tydlig begränsning genom att endast vissa Produkt-use case har beaktats. Vidare är det lämpligt att poängtera att det i dessa vyer ofta kan bli särskilt tydligt att olika

modellerare återger systemet på olika sätt, varvid samsyn inom organisationen är väsentlig vid val av vyer. Ett förhållningssätt som kan rekommenderas är att inte lägga för mycket energi på att finna ”rätt namn” eller att säkerställa att objekten ligger i ”rätt struktur”, bara för att dessa delar råkar vara ens personliga ”skötebarn”. Det viktiga är inte vad saker heter eller var de ligger, utan att de över huvud taget finns återgivna någonstans i modellen. Det är i nedan exempel också tydligt att vad som anses som funktionalitet också kan gränsa till någon form av egenskap som att ta lite plats eller vara kladdfritt, återigen det viktiga är att det skapas samsyn inom den egna organisationen och inte i vilken ”låda” det hanteras.



Figur 5.3 Produktfunktion för stekspaden

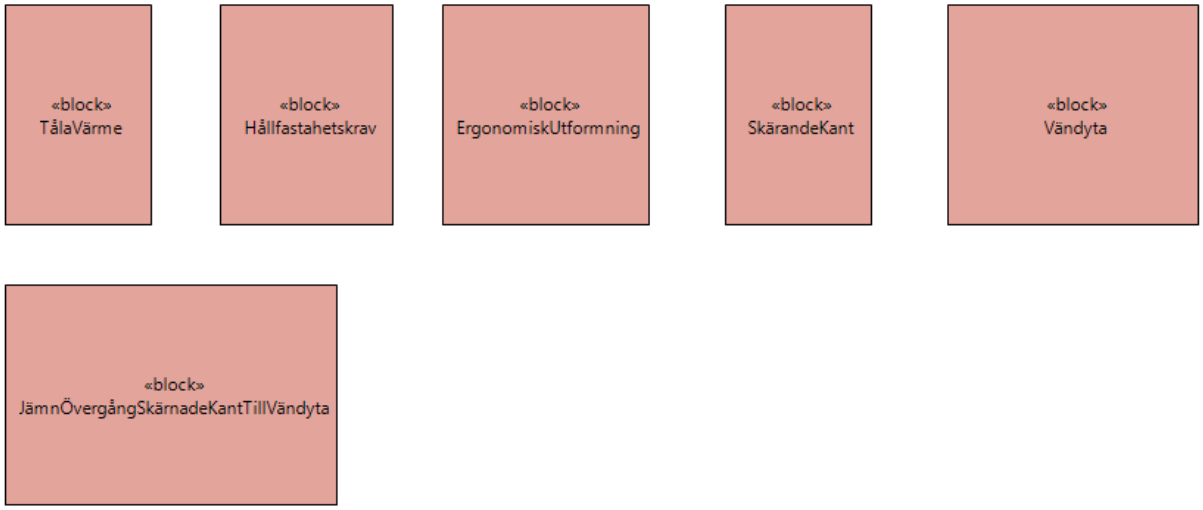
5.6.4 Stekspade: Produkt-use case

För att inte exemplet åter ska svämma över, återges det endast ett litet urval av Produkt-use case baserat på Produktfunktionerna ”flytta mat” och ”dela mat” (figur 5.4). Hur dessa Produkt-use case identifierats framgår av systemvyer, se avsnitt 5.6.6.

Figur 5.4 Produkt-use case

5.6.5 Stekspade: produktkrav

Exempel på funktionalitet och förmågor/egenskaper som stekspaden systemet ska leverera. Nedan visas övergripande några områden som hade behövts kravställas (figur 5.5).

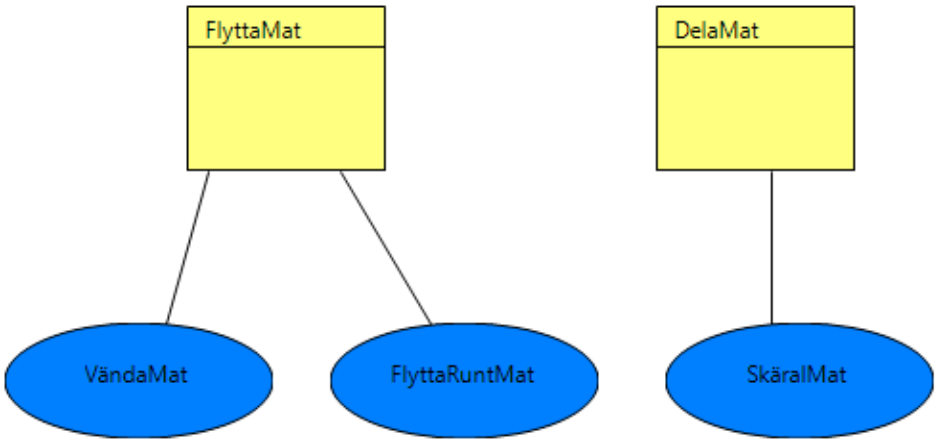


Figur 5.5 Produktkrav för stekspaden

5.6.6 Stekspade: systemvyer produktdefinition

Nedan presenteras den systemvy som blivit mest framträdande för produktdefinitionen av stekspaden. Återigen är det lämpligt att poängtera att dessa exempel är valda för författarnas perspektiv på stekspaden. Systemvyer bör upprättas för att fylla de behov som finns för den egna organisationen och fylla specifika mål och syfte (figur 5.6).

För att återkoppla mot kvalité- och systemsäkerhetsarbete, kan vi tydliggöra med nedan exempel att om det på en sida av stekspaden skulle identifieras en risk, vid en särskild aktivitet, för att den skärande kanten skulle bli slö skulle det i en modell tydlig visa att Produkt-use case ”SkäraIMat” skulle påverkas och därmed också Produktfunktionen ”DelaMat”.



Figur 5.6 Systemvy för koppling mellan produktfunktion och Produkt-use case

6 Arkitekturdefinition

I den tredje abstraktionsnivån är det första gången som systemet explicit närmar sig en realisering. Även om logisk arkitektur inte behöver representera subsystem som faktiskt kommer att realiseras, är det här det som de första explicita designvalen görs för vilka gränssnitt som kommer finnas och hur flöden i systemet kommer att ske. För att kunna arbeta i denna abstraktionsnivå behövs nedan beskrivna följande modellobjekt:

- Logisk arkitektur – regelverk för struktur
- Arkitekturfunktion – redogör för de funktioner som den kommande lösningen behöver för att kunna realisera produktens förväntade funktionalitet under givna förutsättningar.
- Systemaktivitet – är de aktiviteter som systemet förväntas utföra med sin logiska arkitektur.
- Realiseringsarkitektur – produktstruktur med de delar som avser att realisera systemet
- Arkitekturkrav – kravställning

6.1 Logisk arkitektur

Målet med dessa modellobjekt är att visa vilka delar som systemet består av för att kunna leverera produktfunktionalitet. Det är de här objekten som klargör hur systemet kommer att kommunicera, både externt och internt, samt vilka krav som finns mellan olika delar i systemet.

Den logiska arkitekturen syftar till att förklara vad som behövs av systemet. Logisk arkitektur kan därför vara systemdelar som är rent abstrakta och vars enda existensberättigande är att beskriva de ”regler” som behövs för att leverera funktionalitet enligt Produktdefinitionen. Att de är abstrakta är specifikt det som skiljer logisk arkitektur mot design (detaljerad utformning). Logisk arkitektur förklarar uppbyggnad och vad som ska kunna förmedlas över gränssnitt. Objekt för logisk arkitektur blir särskilt tydliga när ett system består av både hårdvara och mjukvara. För ett kylsystem ska fungera behöver det både kunna hantera kylmedium men också ta emot kraft samt sända och ta emot status via elektrisk arkitektur. Vilken typ av (luft/vatten) och vilken typ av kraft (lågspänning/högspänning) som ska kunna hanteras kan lämpligen klargöras i den logiska arkitekturen, medan den rena specifikationen av det konkreta gränssnittet, kablage/kontakt/signal, först klargörs i design (detaljerad utformning). Två exempel på olika arkitekturer som kan behövas för att ett system ska fungera är mekanisk arkitektur och elektrisk arkitektur.

De modelleringsbegreppen som nämnts tidigare i from av black box, grey box och white box kan vara intressanta att nämna i detta sammanhang. Från det övergripande systemets perspektiv kan arkitekturen betraktas som grey box eller till och med white box då det tydligt redogör för delars beroende. För en enskild arkitekturell del i systemet kan dock modellering på denna abstraktionsnivå betraktas som black-box, då den inte behöver gå in på hur kraven från Produktfunktion avses lösas. Precis som för övriga delar i modellering är det viktigt att klargöra vilket perspektiv som modelleras.

6.2 Arkitekturfunktion

Med dessa objekt klargörs funktionalitet som finns kopplad till de olika systemdelarna identifierade som logisk arkitektur utifrån behov i Product-use case.

6.3 Systemaktivitet

Målet med dessa objekt är att klargöra vad som förväntas kunna göras med systemets olika delar på detaljerad nivå.

6.4 Realiseringsarkitektur

Målet med dessa objekt är att klargöra för design vad som behövs för att klara av att utföra systemaktiviteter och uppfylla arkitekturfunktionalitet.

6.5 Arkitekturkrav

Kraven från arkitekturspecifikationen beskriver de styra villkor som respektive delsystem behöver uppfylla för att systemet som helhet ska fungera som avsett. Arkitekturkraven gör det möjligt för utvecklingsprojektet att arbeta vidare med en parallell utformning av alla subsystem, utan att kopplingen till systemet som helhet kommer bort.

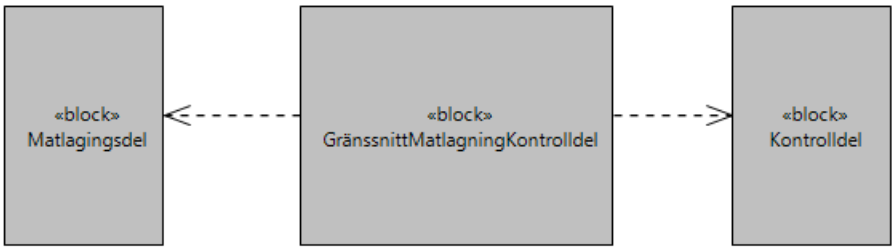
6.6 Exempel stekspade: Arkitekturdefinition

Nedan forsätter exemplet med stekspaden för att visa exempel på de olika modellobjekten.

6.6.1 Stekspade: Logisk arkitektur

Avseende stekspaden fokuserar exemplet på att visa de logiska komponenterna för mekanisk arkitektur. Det blir tydligt att det finns ett behov av att koppla samman hantaget, kontrolldel, med själva spaden såvida de som realisering senare inte är gjutna i en form (figur 6.1).

Ifall den ursprungliga tanken med stekspade varit att den skulle kunna kommunicera samt göra mätningar hade en väsentlig del av modelleringen visat elektrisk arkitektur och kanske även någon form av IT-arkitektur. Det hade varit viktigt att visa hur olika sensorer och kontrolldelar hade organiserats för att lösa ut behoven. IT-arkitektur hade varit extra viktig om vi tänker oss en ”Connected device”.



Figur 6.1 Logisk arkitektur för stekspaden

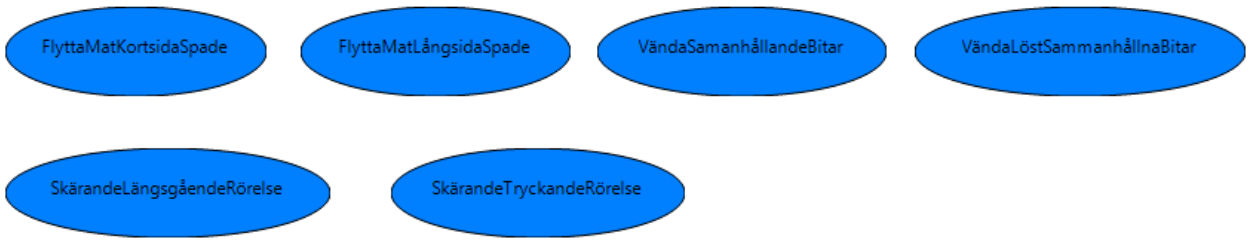
6.6.2 Stekspade: Arkitekturfunktion

Då det för stekspaden fokuserats på mekaniska delar visas i exemplet endast funktionalitet förknippad med den mekaniska arkitekturen. Vidare redogörs bara mathanteringsperspektivet. Detta hade kunnat utvidgas med arkitekturfunktioner från andra områden såsom underhåll, vård eller återvinning (figur 6.2).

Figur 6.2 Arkitekturfunktion för stekspaden

6.6.3 Stekspade: Systemaktiviteter

Systemaktiviteter visar det som förväntas att ske för att klara av att uppfylla arkitekturfunktionerna ovan. Det är viktigt att poängtera att i mekatroniska system kan modeller bli väldigt stora och till viss del komplicerade. Då ställs det särskilda krav för tydlig struktur i modellen. Det är viktigt att klargöra för vilka roller inom organisationen som strukturen byggs. Antingen bryter man ner systemet i mindre delar, tillför fler informationsnivåer som kan hanteras eller så får man hoppas på att man fångat ”rätt” och tillräckligt många aspekter. Det senare är en viktig fråga generellt. Precis som tidigare nämnts så kan man modellera väldigt många olika perspektiv och därför är det viktigt att allt som modelleras har ett mål och syfte för den egna organisationen. Figur 6.3 visar systemaktiviteterna för stekspaden.



Figur 6.3 Systemaktiviteter för stekspaden

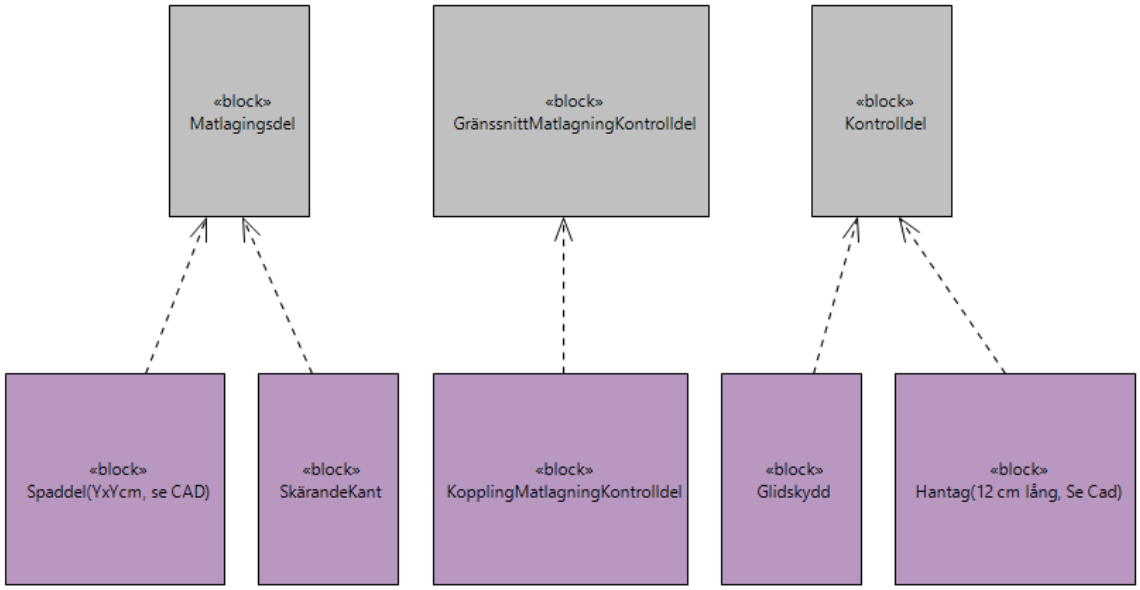
6.6.4 Stekspade: Realiseringsarkitektur

Med dessa objekt visas de delar i systemet som faktiskt kommer att finnas konkret i verkligheten (något man kan ta på) (figur 6.4). Här har vi således kommit så långt att fysiska objekt kan hanteras. De kan kopplas, mätas och kravställas. Det är mellan dessa objekt som alla gränssnittsspecifikationer kommer att behövas för slutgiltig design.

Figur 6.4 Realiseringsarkitektur för stekspaden

6.6.5 Stekspade: systemvyer arkitekturdesign

Nedan exemplifieras på enklaste sätt en nedbrytning från logisk arkitektur till realiseringsarkitektur baserat på de arkitekturfunktioner och systemaktiviteter som identifierats (figur 6.5).



Figur 6.5 Systemvy för kopplingen mellan logisk arkitektur och realiseringsarkitektur

6.6.6 Stekspade: Arkitekturkrav

Kraven från arkitekturspecifikationen beskriver det som den detaljerade designen av respektive systemdel behöver uppfylla. Exempel kan vara friktion på spaddel och handtag, vasshet på skärande kant samt flexibilitet i koppling mellan spaddel och handtag.

7 PU²B-modellen relaterat till andra processer

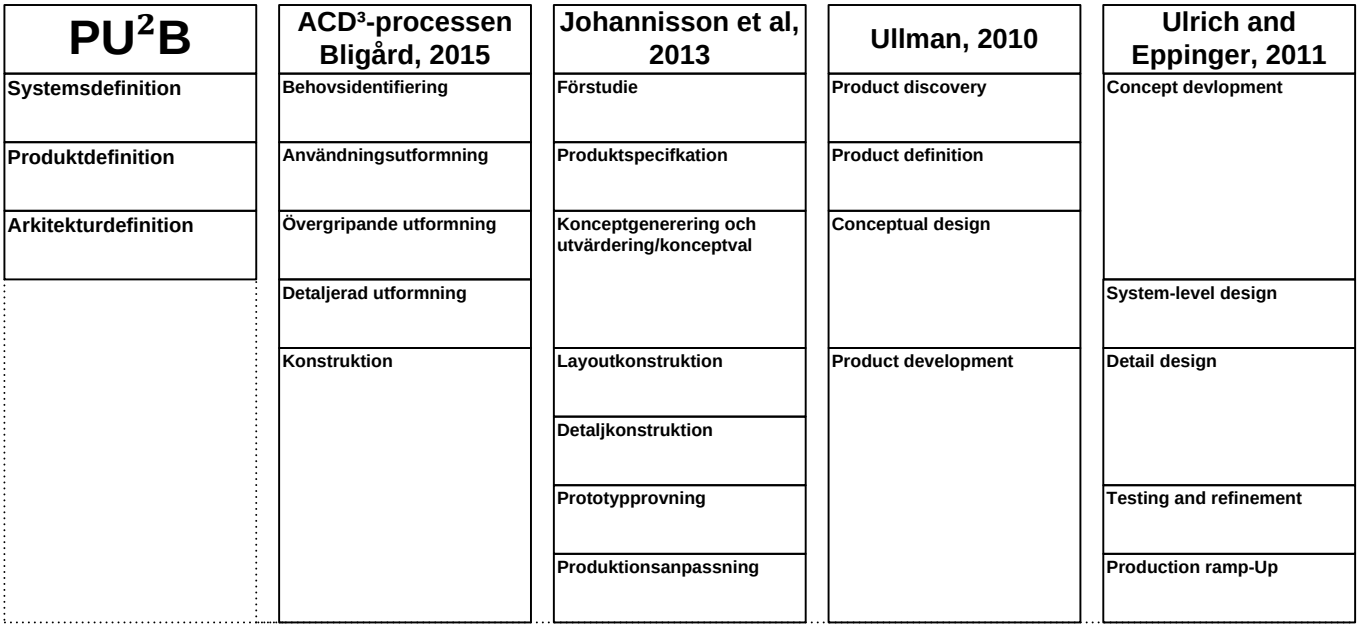
Här behandlas hur PU²B-modellen relaterar till existerande arbetssätt. Kapitlet vänder sig till dem som vill leda och organisera PU²B-modellen i en verksamhet. Kapitlet tar upp:

- Produktutvecklingsprocesser
- ISO/IEC 15288:2008 Systems and software engineering
- OOSEM (Object-Oriented Systems Engineering Method)
- Dubbel-V-modellen (från ACD³-processen)

Kapitlet avslutas med en summering av tankarna kring PU²B-modellen.

7.1 Produktutvecklingsprocesser

Även om nivåerna i PU²B-modellen inte ska ses som steg i en utvecklingsprocess, utan som sätt att beskriva ett system, så går det att koppla dem till faser i utvecklingsprocessen utifrån hur de matchar i fokus. Figuren 7.1 visar att PU²B-modellen i relation till fyra beskrivningar av produktutvecklings-processer (Ullman, 2010; Ulrich och Eppinger, 2011; Johannesson et al., 2013; Bligård, 2015). Modellen svarar upp allstå tydligt mot de inledande faserna i ett produktutvecklingsprojekt, där respektive abstraktionsnivå är i fokus för respektive fas.



Figur 7.1 Relationer mellan PU²B-modellen och fyra olika beskrivningar av utvecklingsprocesser

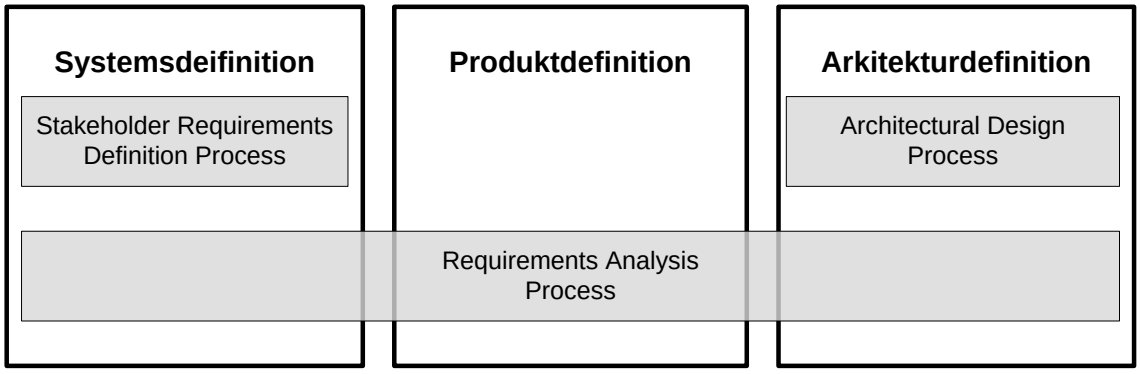
7.2 ISO/IEC 15288:2008 Systems and software engineering -- System life cycle processes

ISO 15288 (ISO, 2008) innehåller ett antal processer och de tre som mest berör PU²B-modellen är finns beskrivna i tabell 7.1: Stakeholder Requirements Definition Process, Requirements Analysis Process och Architectural Design Process. Även här är det de tidiga delarna av utvecklingsarbetet som berörts av PU²B-modellen.

Tabell 7.1 Processdelar i ISO 51288 som främst berör PU²B-modellen

Process part	**Activities**
Stakeholder Requirements Definition Process	Elicit stakeholder requirements Define stakeholder requirements Analyze and maintain stakeholder requirements
Requirements Analysis Process	Define system requirements Analyze and maintain system requirements
Architectural Design Process	Define the architecture Analyze and evaluate the architecture Document and maintain the architecture

Figur 7.2 visar hur processdelar i ISO 51288 är relaterade till PU²B-modellen och den stora skillnaden är att PU²B-modellen, med sina parallella flöden, mer tydligt visar på det iterativa arbetet som sker i ett utvecklingsprojekt. PU²B-modellen lyfter också tydligare fram det designarbete som sker under och mellan Stakeholder Requirements Definition Process, Requirements Analysis Process och Architectural Design Process, vilket benämns produktdefinition.



Figur 7.2. Processdelar från ISO 15288 placerade i PU²B-modellen

7.3 OOSEM (Object-Oriented Systems Engineering Method)

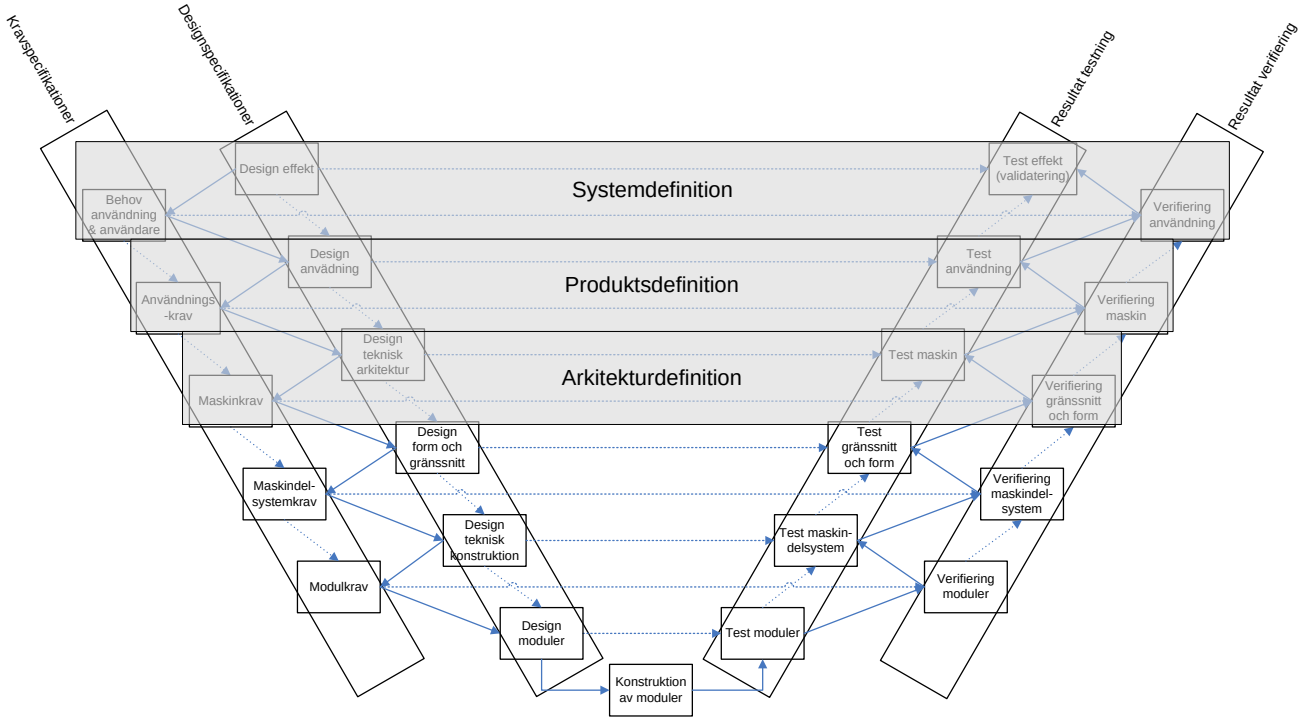
PU²B-modellen matchar väl mot Object-Oriented Systems Engineering Method (OOSEM) (Haskins, 2011). Den stora skillnaden är att PU²B-modellen mer tydligt pekar ut innehållet i de olika processdelarna för OOSEM.

Tabell 7.2 PU²B-modellen jämfört med Object-Oriented Systems Engineering Method och ACD³-processen

PU²B-modellen		**OOSEM process**		**ACD³-processen**
Model object	Purpose	Process part	Activities	Processteg
Capabilities (Customer value)	Identify "selling points", stakeholders and their needs.	Analyse Needs	Causal Analysis Mission UseCase/ Scenarios Enterprise model	Behovs-identifiering
System task	Identify what the system needs to deliver.			
System phase	Identify phases that the system will go through. This will clarify the eco-system and needs of interaction.			
System use case	Identify operator-system and system-system interactions. System boundaries and interfaces will become clearer.			
Product domain object	High level description of one or several concepts for product. Entities assumed to exist and empirical base. A domain model may serve as a first draft of a product structure, later used for implementation.	Define System requirements	System use case/scenarios Elaborated context Requirements diagram	Användnings-utformning
Product function	List desired functionality and functional requirements. Functions may need own Use Cases to explain context of requirements. Functional requirements may need to be extended by non-functional requirements.			
States and modes	In some cases the product States and Modes (refinements of Phases) will make further breakdown of Product Use Cases easier.			
Product use case	Product Use Case explains expectations of Product Functions in detail.			
Logical architecture	Explains how the system is intended to solve functions.	Define Logical Architecture	Logical decomposition Logical scenarios Logical subsystems	Övergripande utformning
System activities	Refinement of Product Use Case to enable detailed design.			
Realisation architecture	Development of Physical architecture identifies interfaces for implementation. This will probably serve as basis for product structure (Compare to Domain model earlier).	Synthesize Allocated Architecture	Node diagram HW, SW, Data architecture System deployment	
Other methods		Optimize and Evaluate Alternative	Parametric diagram Trade study	Detaljerad utformning och konstruktion
Other methods		Validate and Verify System	Test system Test Cases	

7.4 Dubbel-V-modellen

Figuren 7.3 visar hur PU²B-modellens delar matchar mot den dubbla V-modellen ACD³-processen (Bligård, 2015). Den dubbla-V-modellens explicita sampel mellan design och krav i varje nivå gör det enklare att skapa fall för att testa och verifiera tidigt i utvecklingsarbetet.



Figur 7.3 De tre stora delarna i PU²B-modellen markerade i dubbel-V-modellen

7.5 Avslutande summering

PU²B- modellen kan ses som ett nytt sätt att explicit dokumentera det som traditionellt är implicit kunskap. Detta medför att man dels kan testa hur man implementerar företagsstrategi (Business delen) genom att tidigt validera koncept (exempelvis High level MIL). Genom att tydliggöra hur produkten avses användas har det då egentligen redan skrivits grunden för testfall. Vid viss traditionell utveckling skrivs testfall först efter implementering och då måste det som varit ledande för konstruktion identifieras för testning. Genom att använda PU²B- modellen erbjuds en struktur som kan underlätta verifiering och testning, genom att de tydliga ansatserna för exempelvis Produkt-use case kan spegelvändas för test.

Vid en första implementering av PU²B- modellen är det ofta bra att börja försiktigt och i liten skala. Ett första steg är att få en utvecklingsprocess som är uppdelad i nivåer efter abstraktioner och inte bara i nivåer efter ”produktfärdighet”. Viktigt är att se PU²B- modellen som ett verktyg att binda samman verksamheten och inte som ett parallellt spår till befintliga processer. Alltså att PU²B- modellen behöver integreras så mycket som möjligt i existerande utvecklingsprocesser för att den fulla nyttan ska komma till godo. I för hållande till Lean Product Developement är PU²B-modellen ett mycket bra verktyg för att hantera kundvärde och de värdeskapande aktiviteterna för användaren, samt göra utvecklingsprocessen mer naturligt framtung. För Agile systemutveckling sätter PU²B- modellen ramarna för ett agilt arbetssätt. Ett projekt kan övergå i ett agilt arbetssätt när systemarkitekturen är grovt satt.

PU²B- modellen behöver bra kunskap om användaren för att kunna fungera fullt ut. Modellen är alltså ingen ersättning för att jobba användarcentrerat, utan ett verktyg för att få ut nyttan med att arbeta användarcentrerat till hela utvecklingsprojektet. Inom Human Factors så ger området User Experience en bra förutsättning för att förstå vad som är värdeskapande för användaren och inom området Usability Engineering finns många mycket användbara metoder att få ett bra innehåll till systemdefinitionen och produktdefinitionen.

8 Referenser

Bligård, L.-O. (2015). Utvecklingsprocessen ur ett människa-maskinperspektiv - ACD3-procesen. Göteborg, Chalmers tekniska högskola.

Chapanis, A. (1985). Some reflections on progress. Human Factors Society 20th Meeting. Santa Monica CA**:** 1-8.

Flood, R. L. and E. R. Carson (1993). Dealing with complexity : an introduction to the theory and application of systems science. New York, Plenum.

Gustafsson, M. and K. Hedberg (2013). Modell för ökad kunskapsintegrering vid produktutveckling – tillämpning av Model Based System Engineering på Saab. Institutionen för ekonomisk och industriell utveckling. Linköping, Linköpings universitet. **Master thesis**.

Haskins, C., Ed. (2011). Systems engineering handbook - A guide for system life cycle processes and activities, International Council on Systems Engineering (INCOSE).

IEA. (2014). "International ergonomics association web page." Retrieved 2006-04-02, 2006, from http://www.iea.cc/.

INCOSE (2007). Systems Engineering Vision 2020 INCOSE-TP-2004-004-02.

ISO (2008). ISO/IEC 15288:2008 Systems and software engineering -- System life cycle processes, ISO.

Johannesson, H., J.-G. Persson, et al. (2013). Produktutveckling : effektiva metoder för konstruktion och design. Stockholm, Liber.

Karlsson, I. C. M. (1996). User requirements elicitation - A framework for the study of the relation between user and artefact. Göteborg, Chalmers University of Technology. **PhD Thesis**.

Skyttner, L. (2005). General systems theory: problems, perspectives, practice. Singapore, World scientific.

Ullman, D. G. (2010). The Mechanical design process. Boston, McGraw-Hill.

Ulrich, K. T. and S. D. Eppinger (2011). Product design and development. New York, NY, McGraw-Hill/Irwin.

Wickelgren, M. (2005). Engineering emotion : values as means in product development. Göteborg, BAS Publishers, School of Business, Economics and Law, Göteborg University.