Enterprise Design

Här beskrivs också ’the Indigo Equation’, se nedan i löpande text.

Detta är alltså ett sätt att se på olika moment i utveckling generellt och specifikt för digital produktutveckling. Vi har så länge arbetsnamnet ’Indigo Equation’ och det är en modell som adresserar arbetsgång och vilka aktiviteter som behöver göras. Denna bygger på traditionell systemutveckling men praktiskt utifrån hur komplexa problem behöver adresseras. Den har momenten Problemförståelse och definition. Behov, nytta, use-cases. Design av vad som löser detta. Hur man sedan bygger lösningen. Slutprodukten, i detta arbete, ’Informationsvärdering’ praktiskt tillämpbar och arbetsbar.

Vad detta verktyg säger är inget nytt, systemutveckling har mycket länge arbetat så här, det är grunden i agilt och Scrum men detta verktyg tydliggör och breddar att arbeta samtidigt och över alla moment.

Det har ett stort inslag av ’systemteori’ vilket är grundläggande i komplexa problem, med innebörden att alla olika delar i ett problem eller en lösning påverkar varandra, ofta på ett så kompolext sätt att beteenden kan verka oförutsägbara. Detta berör allt inom digital produktutveckling, företag och marknad, team och utveckling, applikationer och användare

Ansats

Det går att designa sitt ’enterprise’, alltså inte bara en verksamhet utan hela organisationen inklusive hur den fungerar ihop med samhället, ’the society’.

Oavsett om:

- ...det är en nu produkt eller tjänst man tar fram, utifrån en insikt om behov hos användare, och då behöver förstå tekniska, kommersiella aspekter inklusive marknadsförning och gränsytor ’touchtpoints’ mot kunder, jämte att den löser behovet.
- ...man istället har en ny produkt eller tjänst eller som skall ersätta en tidigare och det är verksamhetens sätt att använda, exponera, drifta och erbjuda så att hela verksamheten, hela ’enterprise’ skall följa med.
- ...det behövs någon typ av förbättring, av sätt eller kostnader eller andra utifrån identifierade behov och hur det då omfattar alla delar i ’the enterprise’ inkl samhälle och omgivning.

Hypotes och sätt

Sättet att approachera detta är via den framväxande rörelsen kring ’Enterprise Design’. N.b att detta inte är samma som ’Enterprise Architecture’ som delvis har samma syfte men inte heller samma omfattning och bredd. Andra begrepp inom IT-arkitektur som ’verksamhetsarkitektur’, ’lösningsarkitektur’ ingår dock i ’Enterprise Design’ där även ’Design’, ’Designerly way of working’ (se ’Design Way’(Stolterman) ingår, ’Design Thinking’ är lite av samma som ramverken som listas nedan, ett slags ontologi och som i viss mån saknar praktiska verktyg. Vi sammanfattar i vad vi skriver här, arkitektur och design med begreppet ’ArkDes’.

Det finns verktyg i form av taxonomier eller ontologier som hjälper till att förstå och skapa en överblick över den helhet som behövs, men dessa behöver framför allt en metodik kring hur man använder dem. Det är detta vi vill visa hur man kan göra.

- Vi utgår från ’Edgy’, www.entprise.design, som ramverk och dess föregångare ’Enterprise Design Framework’(EDF), https://intersectionbook.com/, och som beskriver denna helhet. Det finns liknande, i grunden utgår alla från Zachman, www.zachman.com eller en förenklad variant från Sundblad&Sundblad, www.sundbladsoftware.com, men även Archimate, www.opengroup.org/archimate-forum/archimate-overview. Men av alla dessa är ’Edgy’ det mest lättillgängliga och praktiska.
- I alla dessa ramverk saknas eller är otydlig, ’the enterprise’ omgivning, främst i form av samhälle, ’the society’. Här får vi gå till byggnadsarkitektur och stadsplanering och en av de bästa källorna är ’Cities for People’(Jan Gehl).
- Med Edgy och ’the Society’ har vi alltså en bild av den helhet, termer och begrepp kring, en ontologi och en förståelse för vad som behövs och kan alltså börja ’designing the enterprise’ med denna utgångspunkt.

Det som sedan behövs är ett arbetssätt.

- Här hämtar vi framför allt sätt att tänka ifrån ’Design Way’(Stolterman) och det som där kallas ’designerly way’.
- Ett sätt att få detta mera konkret är att se det som en ekvation, det är inte en fyrfältare:

Problem definition and understanding
Needs how to Build
---------- = ----------
Designing it the Product
Research

- Detta kallar vi ’the Indigo Equation’ eller mera utvecklad form, ’the Indigo Dyeing Way of Working’ och konceptet hämtas från när man färgar tyg och tygstycket måste bearbetas i sin helhet, neråt, z-led ortogonalt genom alla ytor och ortogonalt och samtidigt, istället för i steg och sekvensiellt.
    - Problem
- Detta sätt att arbeta beskrivs också i boken ’Kreativitet’(Lohman) med exempel från b.la musikindustrin och både hur Mozart och Bruce Springsteen arbetar men andra exempel finns från t.ex Skagenmålarnas olika skisser och detaljstudier på b.la Konstmuseet, Göteborg.
- Man kan alltså utgå från en produkt, antingen som man har eller tänker sig i form av mycket tidiga skisser eller mockups och lära sig om vad problemen och behoven faktiskt är den vägen. Eller man kan gå från ett problem och direkt till produkten man tänker sig och sedan tillbaka till hur den skall designas och byggas.
- Principen är dock att man hela tiden rör sig över hela och alla ytor. Man fastnar alltså inte i en lösning utan måste inse vikten att veta vilket problemet faktiskt är man skall lösa, vad det innebär i funktion och behov och hur det praktiskt skall kunna bygga. Men i skikt med de stora dragen först, fördjupningar där de behövs och att utelämna och inte fastna i det som inte ger värde.
- Att se detta som en ekvatoin istället för en fyrfältare innebär att allt måste balanseras.

Vi ser dock givetvis att det finns en rörelse från problemförståelse, design via hur och att bygga till produkten, och där två bilder kan förtydliga. ’the Sugar Mices’ ifrån Rational Unified Process(RUP) och ’the Design Squiggle’, thedesignsquiggle.com, som båda beskriver rörelsen mot en produkt eller tjänst eller förändring. Dessa illustrerar också en grundtes i

designerly way, som är att det finns en oändlig lösningsrymd medan det i vetenskapliga metodiker handlar om att nå fram till en aboslut sanning.

Sättet att nå fram och leverera ’the Product’ enligt ’the Indigo Way’s ekvation är via ’re-framings’. Alltså att man ser från problem, design, build, produkt ifrån de olika perspektiv som Edgy+Society omfattar. Detta är alltså en mängd olika men det finns vissa huvudgrupperingar. Det finns också en rad olika praktiska verktyg, mallar och andra sätt som beskrivs nedan och kommer bli de huvudsakliga produkter vi tar fram här. Vi ersätter inte befintliga ramverk, metodiker, sätt där sådana redan finns utan utgår istället ifrån genom att hitta praktiska verktyg och sätt som fungerar i verkliga situationer, därför vi ser att det redan finns så mycket bra att arbeta med men som ofta behöver praktisk guidning och en kadens enligt ’mindset – methods – practical tools – build skills för people.’ Men nu adressear vi ’re-framing’ som är sättet att arbeta sig ner och mot ’the Product’.

- Edgy utgör grunden för dessa och handlar om verksamhetens identitet, vad den vill och hur den gör och når ut mot kunder och inåt mot medarbetare. Reframings här kan vara syfte och mål, kontaktytor i form av ’touchpoints’ coh ’channels’ inklusive användares ’situation’.
- Det kan också handla om hur arkitekturen skall utfomras i form av ’capabilities’ som är en grundläggande form och innehåller ’process’ och ’assets’ och som motsvarar de ’use-cases’ och ’scenarios’ som olika användare rör sig i.
- Det kan också handla om ’bridging the gap’ mellan verksamhet och IT, ett klassiskt problem som framför allt hanteras via praktiska sätt inom verksamhetsarkitektur, user story mapping, att förstå domän med terminologi. Men framför allt utifrån grundtesen att ’use cases’ måste motsvaras av funktionalitet, ’architectural form’ och som ger grunden för implementation ’architectural structure’, vilket är inspirerat av ’Lean Architecture’(Coplien) och där vi redan har en produkt ’ABCDesEntOrgSociety’-pattern vilket även adresserar materialet produkter byggs av, digitalisering och är en praktisk implementation av ’software engineering’.

Sättet att närma sig och nå fram till ’the Product’ är alltså att förstå vad problem och behov är, hur man kan designa och sedan bygga och praktiskt hela tiden ta fram produkten, enligt ’the Indigo Equation’ och i form av olika re-framings.

- Praktiska re-framings finns i en mängd olika former och hämtade från designrörelsen, IT-arkitektur, affärs och produktutveckling, traditionell byggnadsarkitektur, den agila och lean-rörelsen.
- Vi nämner här bara några men här kommer vi som en produkt lista fungerande, praktiska sådana eller hur man får bra metodiker praktiskt användbara och i den helhet som detta utforskande genom alla ytor behöver.
- Informationsarkitektur och informatoinsmodeller. User Scenarios och User Stories och Use Cases. Förmåge och Processorientering. M.fl.
- Koncept som ’viability, feasability and usability’ eller utifrån och som blir en annan framing, ’desireability’ och tankar ifrån den sociala dimensionen (Patrick Jordan).

Från industridesign

Sammantaget kan man säga att det handlar om ’organise the product in its entirety’(Dieter Rams).

- ’Entirety’. Genom att ha en helhet i form av en ontologi och Edgy som ger aspekter och ytor.
- ’Organise’. Att genom reframings som bygger på alla olika aspekter och ytor ’organise’ genom att fördjupa förståelse, lära tillsammans, prova genom experiment och konkret bygga på produkten.

Från agila rörelsen

Ytterligare en del är ’att bygga kunskap gemensamt’

- Att det finns den tydliga gräns mot projekt och designsituationen
- Att denna gräns är navigerbar genom sätt som demos, tydliga leverabler, att olika intressentere kan dras in i den kreativa loopen, ’the creative density’ via plats och ’dialogväggar’ som är ännu en produkt vi har.
- Här hämtar vi också inspiration och kan använda eller översätta praktiska verktyg direkt mer eller mindre från den agila och designrörelsen.

Det måste vara praktiskt

Praktiska verktyg har vi redan adresserat men det är ännu ett huvudproblem med många olika metodiker och ramverk som lärs ut eller tillhandahålls av konsultbyråer. Detta problem är att dessa oftast demonstreras, lärs ut via mycket förenklade exempel i kombination med att praktiskt verktyg ofta saknas.

- Dessa verktyg som praktiskt fungerar i kompelxitet, är överkomliga i pris och resurser som roller som ha ’skill’ och tid. De måste också kunna hantera den stora komplexitet som ofta även små arbeten innebär.
- De produkter vi kommer tillhandahålla här adresserar detta på flera sätt. Som mallar och tillägg till befintliga verktyg som Visio eller draw.io. Som praktiska implementationier av ’mechanics’ hämtade ifrån brädspel och gaming/gamification. Som framför allt nu inledningsvis, i form av att adressera enligt mindset ’det går alltid att informera, ofta mildra, ibland lösa’ och koncept ifrån klassisk grekiskt tänkande kring kunskapsformer som teori, praktik och praktisk kunskap, ’phronesis’ vilket utformades som enkla ’tänkespråk’ b.la inom tidig kristen tradition, ’Apophthegmata’ eller ’Tänkespråk’. I en mera modern form kan detta utformas som illustrerande, belysande, enkelt tillgängliga och ständigt närvarande, ’ ubiquitous’.

Ytor – the Indigo Equation

Denna utgår, inte från aktiviteter utan ifrån vad digital produktutveckling består av. De olika areor, eller ytor eller spaces man behöver röra sig i och utföra arbetet. Hur dessa hänger samman, alla är likvärdiga och behöver kontinuerligt adresseras.

Problemförståelse

’Problemförståelse’ är en separat aktivitet eller yta och som ofta nedprioriteras. Denna sträcker sig från syfte i ett projektdirektiv till vad det faktiskt är en användare behöver eller vad en funktion faktiskt skall vara och utföra.

- Här hämtar vi koncept ifrån flera olika håll men det grundläggande är alltid att man inser att det som ofta kallas ’vision’ eller ’syfte och mål’ också ofta utformas som luddiga statements på få rader. Det kan också heta ’Nyttonalays’ vilket också ofta är en aktivitet

som ger bra insikter men saknar ofta praktisk metodik att följa med och utgöra testfall i slutändan eller strategisk uppföljning.
- ’Conceptual Blockbusting’(Adams) är en bra beskrivning.
- ’the Cynefin Framework’(Snowden) är en annan bra beskrivning.
- Båda bygger på insikten att problem kan vara enkla, komplicerade, vilket kräver utredning eller komplexa, vilket enbart kan lösas med utforskande, probe/sense. Även här har vi en konfliktyta med traditionell projektmetodik som normalt inte adresserar komplexa problem.

Problemförståelse börjar i Research som alltså är en annan av de generiska aktiviteterna, som ingår i alla olika ytor. Alltså, för att förstå ’Needs’ måste man givetvis göra research, förstå vad behoven är men också vilka användare och andra roller som finns och vad de har för respektive behov och (tbd men en mall med olika idesporrar för vilka aktiviteter och leverabler och aspekter som behövs här).

Research

Vi vill också lyfta research som en generell yta eller aktivitet och som behövs för olika delar, ’Need’, ’Design’, ’Build’ och ’Product. För att förstå Problem behöver man givetvis research men detta behövs också kring Design, hur man ’Build’ och när ’Product’ är klar, så att den uppfyller behoven och fungerar tekniskt.

Designsituationen

Ett av de andra huvudproblemen är att det vi kallar ’the design situation’ där vi sysslar med ’enterprise design’ behöver fungera gentemot de projekt som normalt alltid driver arbetet. Projekt är traditionellt sekventiella och baserade på vetenskapligt tänkande och detta kan bli en konflikt både kring frihet att arbeta arkitekturellt och designerly och i leverabler som tidsuppskattningar, prioriteringar, sätt att se på status. Ett sätt att hantera detta är att skapa en ’designsituation’ som är skild med tydliga gränser emot, ’projektsituationen’. ’Design, Process och Metod’(Wikberg Nilsson) och ’Design Things’(A.Telier) är metodiker vi utgår ifrån och som beskrivs praktiskt i en annan av våra produkter ’DesignSituation Plastboxen’ med ’AlltSomBehöverVara’ och ’AlltSomBehöverBli’ som huvudsakliga steg jämte ’Produkten’ och ’Verkstaden’.

För att hitta arbetssätt så hämtar vi inspiration ifrån flera källor.

- Den agila rörelsen givetvis. Här är dock viktigt att notera att denna ofta hamnar i situationer där målet är ständiga förbättringar, funktionstillväxt, DevOps och via backlog-drivet arbete i form av Scrum eller Kanban. Här kan agilt tänkande hamna lite fel när det möter ’designerly’ arbetssätt och vi försöker vara tydliga i detta.
- För att adressera detta, ’The Lean Startup’(Eric Ries) och även praktiska metodiker från Kromatics, kromatics.com.
- ’Product Discovery’(Torres) och i bredare koncept från ’Tolpagorni’, tolpagorni.com.

Human dynamics och Engagement model

Detta är aspekter på IT-arkitektens roll ifrån www.iasa.org tidigare kunskapsmodell, ITABOK, som nu är delvis omgjord. Men det handlar om hur man hanterar arbetssituationer, roller och personer.

Iasa beskriver ’Engagement Model’ i sin BTABOK: https://iasa-global.github.io/btabok/engagement.html. Från denna kan man också hämta förståelse kring ’Problemförståelse’ t.ex genom JTD-ramverket, se en bit ner på sidan, där olika aktiviteter som behöver göras listas, i form av ’Main jobs to be done’ och ’Related jobs to be done’. Med underaktiviteter. Just detta är en bra ingång till förståelse av hur arbete med leverabler av olika typ behöver anpassas, och alltså innebär ’related jobs’ som presentation för intressenter eller hur användare skall förstå en design som behöver critique.

Hur hitta olika Arbetsdokument för olika roller

Utifrån ’the Game Board’-version som visas i _100-modellen så behöver man ha en arbetsyta, någon form där man kan arbeta med information. Denna kan vara komplex, ha enkla strukturer, innehålla mycket och olika information för att skapa en kreativ densitet, för teamet.

För intressenter, användare och andra grupper behöver dock denna vara i annat format. Från arbetsdokumenten, ytorna behöver man sammanställa och publicera leverabler som är begripliga, tillgängliga(inte verktyg som dessa inte har licens för) och möjliga att samarbeta kring.

Exempel på verktyg:

- Visio eller draw.io, grafiska verktyg
- En ’design.log’ med både olika stories att arbeta med och en log kring vad som händer i arbetet. Te.x Word.
- En research-yta. T.ex ’Obsidian’ eller Visio/draw.io-modell I form av ’Periodiska systemet’

Dessa olika verktyg beskrivs i _1nn ’Verktyg’(tbd)

Praktiskt arbetssätt

Så hur gör vi i en konkret situation när vi skall ta oss igenom ’the Indigo Equation’ och börja fylla i ’ABCDEsEntOrg’-design pattern. Det är dessa två huvudsakliga platser eller leverabler eller verktyg vi rör oss kring. Utifrån dessa två sammanfattas mycket av arbete i repsektive metodiker som olika verksamheter kan ha oavsett om det är designdrivet, tjänstedesign och Service Blueprints eller andra metodiker. Eller arkitektur och ramverk som TOGAF eller ontologier som Zachman eller kombinationer som Edgy och ’Enterprise Design Framework’.

Inför planering av en workshop eller mera löpande arbete i en egen roll eller i ett team är att man i sitt arbete kan tänka utifrån:

Design Patterns som en ’Stadsplan’ o som ofta är hur arkitektur börjar. Andra former är Vintergatan eller att man väljer rutor ifrån Zachman. Eller utgår från en mera helhetsbild som Edgy. Det blir ofta så att olika metodiekr föreskriver att t.ex ’Börja med en Stadsplan’ men man behöver ställa sig ett antal frågor menar vi.

- Vad är det vi skall göra? Är vi i en mera utforskande fas. En beskrivning av ett läge isåfall för status eller för att driva arbetet vidare? Eller är vi i en utvecklande fas? Det beror alltså på läge i arbetet. Det kan också vara mera dokumentation och det rör sig om att förbereda för förvaltnings och systemdokumentation. En rad olika frågor finns här.

- Sedan för vem? Form, språkbruk, termer, perspektiv/framings beror på detta. Är målgruppen de som skall köpa tjänsten, eller är det intressenter, en styrgrupp eller olika medarbetare eller slutanvändare.
- Sedan perspektiv/framings som är en egen fråga. Här kan man utgå ifrån Edgy, men EDF eller Zachman är liknande. Men alltså Verksamhet eller styrning eller användares outcome, mål och resan dit via olika kanaler och touchpoinst. Architektur o vad det kan vara, mera funktionerna o förmågorna. Eller Experience o här är vi närmare Users-elementen i mitten i form av slutanvändare. Users-elelmenten är ju i mitten därför att det finns users även runt Identitty o i Architecture som t.ex 'tjänstens medarbetare'.
    - Det går att kombinera ett par, några men behövs annars flera olika.
- Sedan vilken är konversationen vi vill ha? Är det förståelse för vad produkten är? Vad projektet gjort? Vad produkten i förlängningen skall bli när projektet är klart?
- Sedan form. Text. Bild. Storytell. PDF eller Word eller Markdown. En kombination? Detta beror också på hur och var det skall publiceras. En samarbetsyta, repo, som dokumentation i mera av ett bibliotek. Det har med tillgängighet och samarbetsmekanismer, formalia som versionshantering eller annan metadata att göra också. Sekretess och rättigheter.

Att minimera variation – double loop learning

Det finns inte bara ett sätt att arbeta, inte bara en uppsättning metodiker och verktyg och varje designsituation har sina egna förutsättningar.

Att lära sig hantera, att tänka i hur olika sätt behövs för att möta dessa olika situatoiner handlar om learning och att få detta kontinuerligt, en 'learning loop'. Om man går ett steg vidare och kan hantera, tänka i hur man lär sig och hela tiden förbättrar denna lärande loop så har man tagit arbetet ett steg itll, 'double loop learning'.

Man kan se dessa som att a designerly way-arbete liknar 'the Design Squiggle'. Genom att hantera detta, om man tänker i en första derivata, så kan denna tvådimensionella rörelse reduceras till en dimension där man hanterar variation i hastighet och genom att hantera även detta, en i det närmaste rak linje.

De olika sätt och metodiker som beskrivs här syftar till att etablera denna double loop learning, det är vad verksamheter behöver göra. Olika sätt finns sedan att hämta utifrån situation och behov, vi listar en rad olika och kommer komplettera både med egna och framför allt med andra men beskrivna praktiskt och hur de kan samverka. Tillhandahållare av olika metodiker brukar sällan visa hur de kan kombineras med metodiker från andra, det tror vi är ett gap som behöver hnateras. Verksamheter behöver alltså hela tiden anpassa till vad som fungerar i en given situation och därför finns inte en färdig kombination av olika sätt beskrivna här utan det vi vill beskriva är hur man hanterar och skapar en double loop learning-situation och uppmana till att hela tiden fråga sig, vad behöver vi lära oss nu, inte bara gällande produktens utveckling utan även arbetssätt.