

## **Business Process Management (BPM) Agile ?**

Et af argumenterne for anvendelse af BPM er at man opnår en agile løsning. På dansk også kaldet adræt løsning. Med dette menes at BPM som koncept fører til løsninger som straks levere værdi til kunden gennem iterativ udvikling. Med en agile løsning kan vi levere IT løsninger hurtigere og med højere kvalitet som svar på forretningens ønsker og krav gennem iterativ modellering. I agile løsninger taler vi om en højere grad af flexibilitet og effektiv udnyttelse af ressourcer hvor vi gennem iterationer drager fordel af sidste iteration og korrigerer fejl. Vi tilpasser processen til nye krav og genbruger alt fra sidste iteration. Typisk vil forandringer af processen kunne designes på timer og implementeringen af processen kan ske på dage. BPMN modeller er samtidigt modul uafhængige og vil typisk involvere mere end et modul i SAP.

### **Business Process Modelling**

Måden hvorpå dette kan lade sig gøre er, at se på IT løsningerne på en ny måde. I stedet for at se processen som én samlet IT-løsning, bør vi adskille den forretningsmæssige proces fra den implementerede tekniske løsning. Ser vi på processerne med forretnings øjne kan vi enkelt abstrahere fra de tekniske detaljer og dermed kan vi opbygge eller forandre processer på ganske få timer. Som værktøj anvendes grafiske værktøjer der understøtter Business Process Modelling Notation (BPMN), et værktøj med hvilket vi kan beskrive en proces grafisk og hvor vi kan modellere sekvensen af aktiviteter, beslutninger og konditioner. Hvis vi forstår at abstrahere fra detaljer i processen som inddata, resultat data, bruger grænseflade mm, kan forretningen sammensætte processer næsten uden IT afdelingens medvirken. I denne del af udviklingen kan man virkelig hurtigt opnå den agile løsning.

### **Business Process Execution**

Hvis vi kan acceptere påstanden om, at virksomheden kan drage fordel af BPMN modeller og at virksomheden enkelt på ganske få dage, kan designe sine forretnings processer, vil det være naturligt at gå næste skridt også. Hvis virksomheden kan afvikle sine forretningsmæssige processer via et process execution værktøj, så vil forretningen kunne forandre og afvikle de forretningsmæssige processer uden hjælp fra IT afdelingen. I det grafiske modellerings værktøj bygger på BPMN som kun indeholder ganske simple regler, er det forholdsvis enkelt at udvikle en

process execution engine som kan afvikle BPMN modeller. I det BPMN er en standard indeholder BPMN alle de regler som vores process engine skal anvende. myProcess Manager er et eksempel på en sådan process engine. Alternativt kan man anvende SAP's Business Workflow eller SAP Netweaver CE med BPMN.

### **Business Process Monitoring og BI**

Når nu vi har accepteret BPMN som model og har udviklet et process engine, der kan afvikle processen, hvad er så mere nærliggende end at udvikle et overvågnings værktøj i hvilket vi kan følge op på vores forretningsmæssige processer. I princippet er det jo ganske ligetil at måle om processen er startet, hvornår den startede, hvornår den sluttede, hvor lang tid den kørte samt hvilken status processen endte op med. Endvidere kunne vi passende logge alle evt. fejl og system meddelelser fra processen et og samme sted f.eks i applikations loggen. Derved har vi data grundlaget for overvågning og fremstilling af BI-rapporter. Rapporter og analyser af overvågnings data skal anvendes som grundlag, for den næste iterative udvikling af processen. Med overvågnings data kan vi netop identificere de dele af processen som enten indeholder fejl, ofte er skyld i ringe performance eller ofte er udsat for forandringer. Dermed kan vi holde fokus på de dele af processen hvor det virkelig kan betale sig at forandre.

### **Business Process Implementation**

Forudsætningen for en succesfuld BPM løsning er at IT afdelingen understøtter forretnings processerne med kode fragmenter som umiddelbart lader sig genbruge. I IT afdelingen skal vi derfor kunne producere kode fragmenter som kan lægges på lager og som kan genbruges i forskellige sammenhænge. I IT afdelingen skal vi mere tænke i komponent løsninger end i samlede og forkromede IT løsninger. Men hvordan skal IT afdelingen som jo netop skal udvikle detaljer, kunne abstrahere fra de tekniske detaljer.

Løsningen er ikke helt så enkel endda og det kræver en stor indsats i IT afdelingen. Det er ikke noget som sker lige med det samme, det tager tid, men det er indsatsen værd. Løsningen er at udvikle sine løsninger med ABAP Objects og design patterns. At genbruge alt hvad vi selv har udviklet og alt det SAP har udviklet. Vi skal betragte SAP systemet og vores egne løsninger som et sæt af services der står til rådighed, vi vælger dog selv hvordan vi vil anvende disse services.

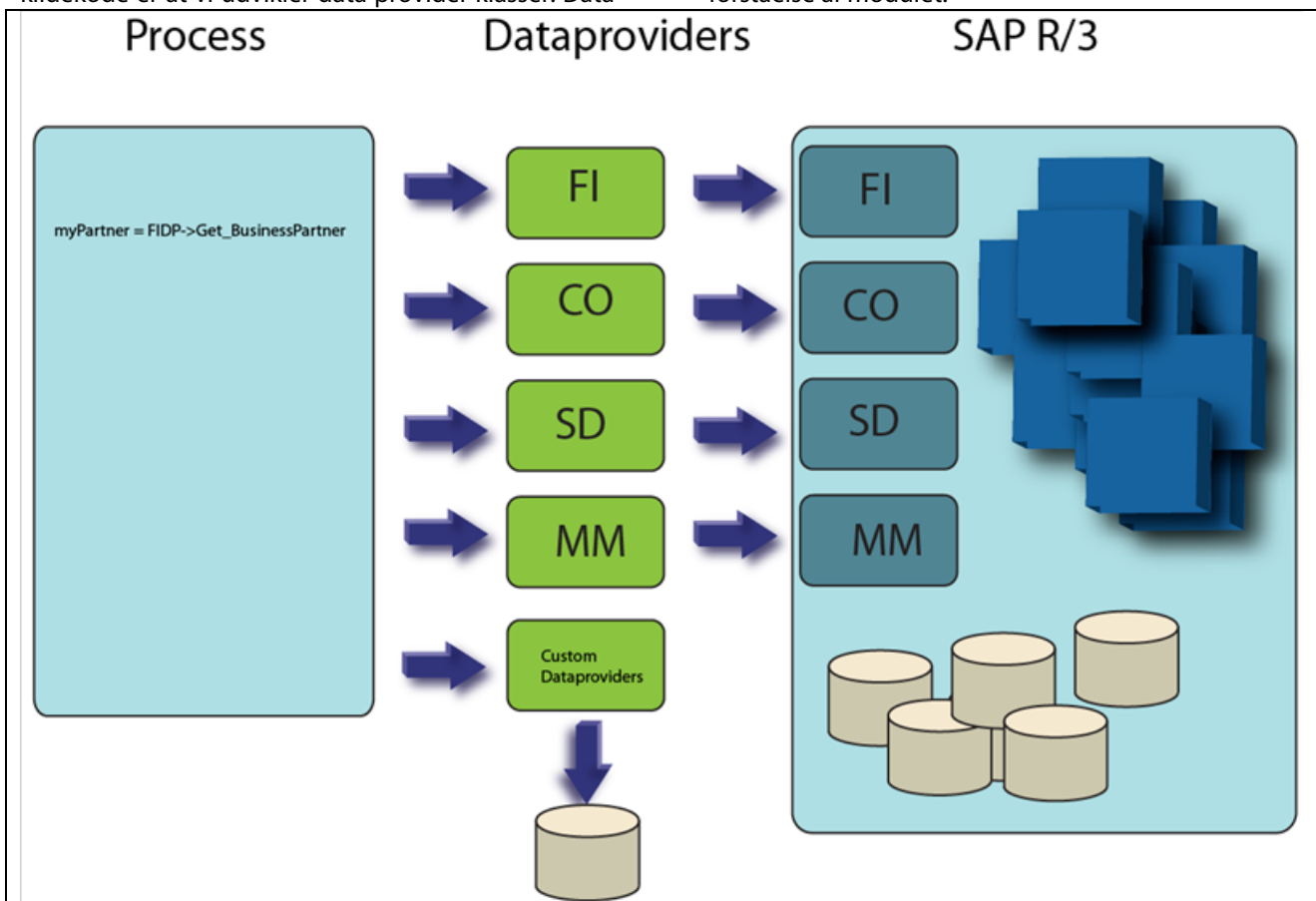
Inden vi går til detaljerne skal vi først se lidt på design patterns. Design Patterns er objekt orienterede løsnings modeller der er sproguafhængig. Om vi anvender C++, Delphi, PHP eller ABAP er i princippet ligegyldigt. Med design patterns har IT afdelingen et fælles sprog til beskrivelse af modeller og skabeloner for løsninger. Design Patterns findes vel beskrevet på internettet. Design Patterns hjælper løsnings arkitekter til at fokusere rigtigt og til at kommunikere løsnings designet videre til programmørerne.

Nogle af de designpatterns som et projekt meget hurtigt får brug for er factory, singleton, adapter, composite, facade, decorator, proxy. Når standard design patterns kommer ind under huden på løsnings arkitekterne opstår behovet for udvikle DATA PROVIDERS og her kommer vi så til det essentielle. Grundlaget for den høje reduktion i kildekode er at vi udvikler data provider klasser. Data

providers er normalt ikke en del af de design patterns som man finder beskrevet i litteraturen. Det skyldes at data providers er tæt knyttet til de miljøer i hvilket de skal eksistere.

Grundlæggende er data provider en klasse som indeholder metoder, der kan anvendes til at hente data fra forskellige kilder. Vi kan anskue data providers som et interface eller grænse flade mellem vores process og SAP eller egne udviklede add on moduler. Ofte vil man i et SAP system udvikle en data provider pr. SAP modul.

Men er det ikke i strid med BPM konceptet, som netop går på tværs af moduler. Næh egentlig ikke. I proces modellen kan vi forholde os abstrakt til systemet, hvor vi i data providers skal forholde os til detaljerne. Detaljer kræver ofte en tilbundsgående forståelse af modulet.



I figuren ses at vores process implementering indeholder simple kald til data providers. Her skal vi i processens aktivitet have fat i forretnings partnerens identifikation. Processens aktivitet kalder derfor den retmæssige data provider, her vores FI data provider. Data providers bør fungere som on demand metoder, hvilket betyder at klassens metode skal holde styr på

om klassen allerede har den nødvendige information, hvis ikke så kaldes standard SAP klasse metode eller funktionsmodul som kan levere de nødvendige data. Standard SAP indeholder mange løsninger som hver især kan løse et og samme problem. Vores opgave er således at finde den bedst tænkelige generelle løsning. Det er ikke sikkert at den første løsning er den bedste, men blot vi sikrer at input/output

parametre til metoden forbliver uforandret, så kan vi med stor succes senere reimplementere kode uden at det påvirker eksisterende løsninger negativt.

Dataene gemmes bla. i data provider klassen så de netop kun læses én gang i data providers livscyclus. At data provider klassen er on demandi betyder at udvikleren af processen ikke behøver tænke over om data er tilgængelige eller ej, i processen kalder man blot data provider metoden og forventer at få returneret de korrekte data. Om processen er startet eller genstartet skal for processen være ligegyldigt. Denne detalje skal være indkapslet i data provider og process manager.

I SAP systemer ses ofte mere end 10 måder, at løse samme problem på. Årsagen er både historisk og de mange forskellige ressourcer som gennem tiden har arbejdet på systemet. Med Data providers løser vi denne problemstilling, idet enhver programmør skal lære kun at anvende data providers når data skal hentes. I starten hvor data providers indføres, er det selvfølgelig vanskeligt og kræver ofte at data provider udvides med ny funktionalitet. Men på sigt opnår man at der kun er én metode til at hente f.eks forretnings partners id på. Det betyder, at vi uden at forstyrre igangværende projekter og projekter i produktion, kan optimere og ændre implementeringen af vores data providers. På sigt giver det mere stabile og hurtigere metoder. I forbindelse med en opgradering, vil det kun være disse interfaces mellem SAP og egen udviklede løsninger der skal opgraderes.

Men hvorfor har vi så ikke bare altid gjort dette, hvorfor er det først nu vi virkelig sætter fokus på genbrug. Der har såmænd været fokus på det før, idet løsnings arkitekter har plæderet for, at samle funktionalitet i funktionsgrupper. Men det først med kundernes accept og anvendelse af abap objects, at det er praktisk muligt at gennemføre.

I starten af et projekt der anvender data providers, er det svært, idet programmørerne ikke længere bør kalde standard SAP funktionsmoduler, BAPI, Objects, programmer eller anden funktionalitet. Programmørerne bør heller ikke udvikle SQL statements i de enkelte proces implementeringer. Hver gang der er brug for en standard funktion eller læsning af data med SQL, bør man stoppe op og skrive koden som en metode i en data provider klasse. Siden hen er det enkelt at genbruge ligesom man enkelt kan reimplementere koden eller performance optimere. Men skal projektet have succes's, skal man være pragmatisk og tillade både

kald af funktionsmoduler og egen udviklet SQL, men det bør begrænses mest muligt.

Det allerbedste ved at anvende design patterns er, at de kan anvendes i mange forskellige sammenhænge. Det er ikke kun til BPM løsninger, men også til mere klassiske løsninger, rapporter og selv til SAP Business Workflow.

I denne artikel har jeg taget udgangspunkt i BPM og kun omtalt Data providers. Men i den fuldendte løsning vil man bygge andre design patterns klasser som f.eks en SystemChange klasse som indkapsler standard SAP funktionalitet for opdatering i SAP. Validation Manager klasser for valideringer, Exchange klasser for data udveksling, service klasser for publicering af services, der stilles til rådighed for web løsninger og andre del systemer og hertil kommer mange andre klasser som alle har til formål at indkapsle funktionalitet, så vi opnår en komponent orienteret verden med komponenter der kan genanvendes i mange forskellige sammenhænge.

I min pragmatiske forståelse af BPM kan en BPM applikation opdeles således

### **1. Business Process or the Business Logic layer**

- a. Modellering af forretnings processerne
- b. Konfigurering af processerne i Process Manager
- c. Test of the konfigurerede processer (før udvikling starter)

### **2. Business Process design & Implementation**

- a. applikationens klasse hierarki designes, muligvis med tomme klasser, altså uden metoder. Klasse hierarkiet videreudvikles under hele applikationens livsforløb.
- b. Udvikling af Process Execution Classes, altså implementeringen af de enkelte processer. Dette er en iterativ process og det er vigtigt at starte med de simpleste processer først.
- c. Løbende udvikling af de klasser som understøtter processerne f.eks data providers, validation managers, system change, exchange managers og mange andre.
- d. Udvikling af de triggere som skal kunne igangsætte eller genkøre processer. Her er det afgørende hvilken applikation der udvikles, hvorfra kommer data og hvad der skal ske med de databehandlinger som er foretaget. Skal processer kunne startes automatisk, via et bruger interface, via events, online, batch mm.
- e. Udvikling af bruger grænse fladen, specielt mod Process Document og det standard SAP bilag som er den primære databære f.eks SD Bilag, FI Bilag osv.
- f. Udvikling af grænseflade mod systemet, herunder tekniske detaljer som kommunikation, database og udvikling af data providers, system change.

### **3. Business Process Execution layer**

- a. Business process execution
- b. Business process monitoring
- c. BI & reporting

Erfaring fra tidligere projekter viser, at adskillelsen mellem den forretnings mæssige applikation og den tekniske implementerede applikation er vigtig og vil bestå. Men erfaringen viser også at grænserne flytter sig i retning af, at forretningen i højere grad får direkte indflydelse på processernes implementering. Har IT afdelingen udviklet alle de nødvendige klassemetoder, skal der ikke megen kode til for at en process vil fungere.

