Note om Enterprise Integration

Information Integration Architecture

Information og data er centralt i ethvert integrationsprojekt. I sidste ende handler integration om udveksling af forskellig repræsentation og typer af data mellem systemer. Den udfordring der ligger til grund for alle integrationsprojekter, er, hvordan man kan muliggøre interoperabilitet mellem systemer med data i forskellige formater og strukturer.

Information Integration Architecture definerer infrastrukturer og processer der muliggør at information kan være tilgængelig på tværs af systemer og platforme.

Løsningen på dette er at data repræsentere i et kanonisk dataformat (Canonical Dataformat).

Det kanoniske dataformat øger i høj grad genanvendeligheden og reducerer både implementerings-/operationelle omkostninger og tid.

Hvert system skal kun mappes til det kanoniske format, så kan det fungere sammen med andre systemer i samme kanoniske format.

Det er grunden til, at JSON og XML er så vigtige i dag. Den udbredte accept af JSON og XML skyldes i høj grad det enorme behov for at beskrive data i et fælles format for dermed at reducere tid og omkostninger ved integration.

Men selvom JSON og XML begge indeholder et standardiseret kanonisk format, så afhænger dataenes reelle værdi af, at dataenes integritet bevares på tværs af systemer. Behandles de samme data i flere forskellige systemer, så er det vigtigt, at værdien af disse data er ens i alle systemer.

Løsningen til at sikre at værdien, betydningen og integriteten af data på tværs af systemer er Metadata. Metadata er information om data.

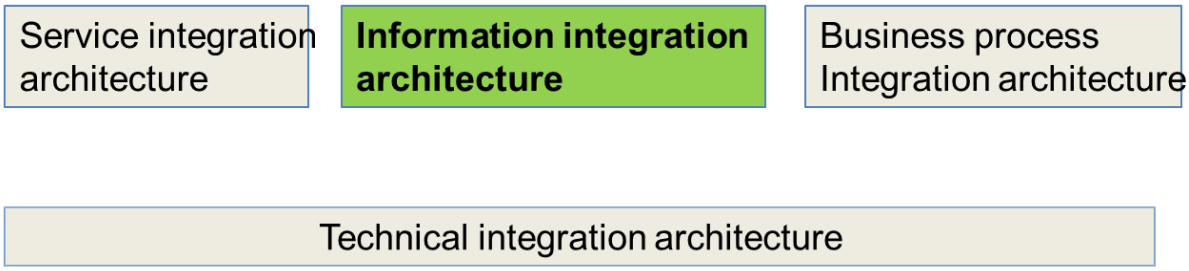
Jo mere beskrivende, præcis og komplet beskrivelse af data vi har, jo bedre vil integrationen kunne implementeres.

Mens der måske allerede findes en stor del af metadatabeskrivelser i systemer, er der en række værktøjer på markedet, der automatisk kan udtrække metadata fra kildesystemer. En forudsætning for kvaliteten af disse udtrukne metadata afhænger dog af mængden og kvaliteten af de metadata, der eksisterer i kildesystemerne. For at opnå det fulde udbytte af metadata skal virksomheden sikre, at metadataene er nøjagtige og komplette. Det vil kræve betydelige investeringer. Men på bundlinjen vil den investering give gevinst på sigt.

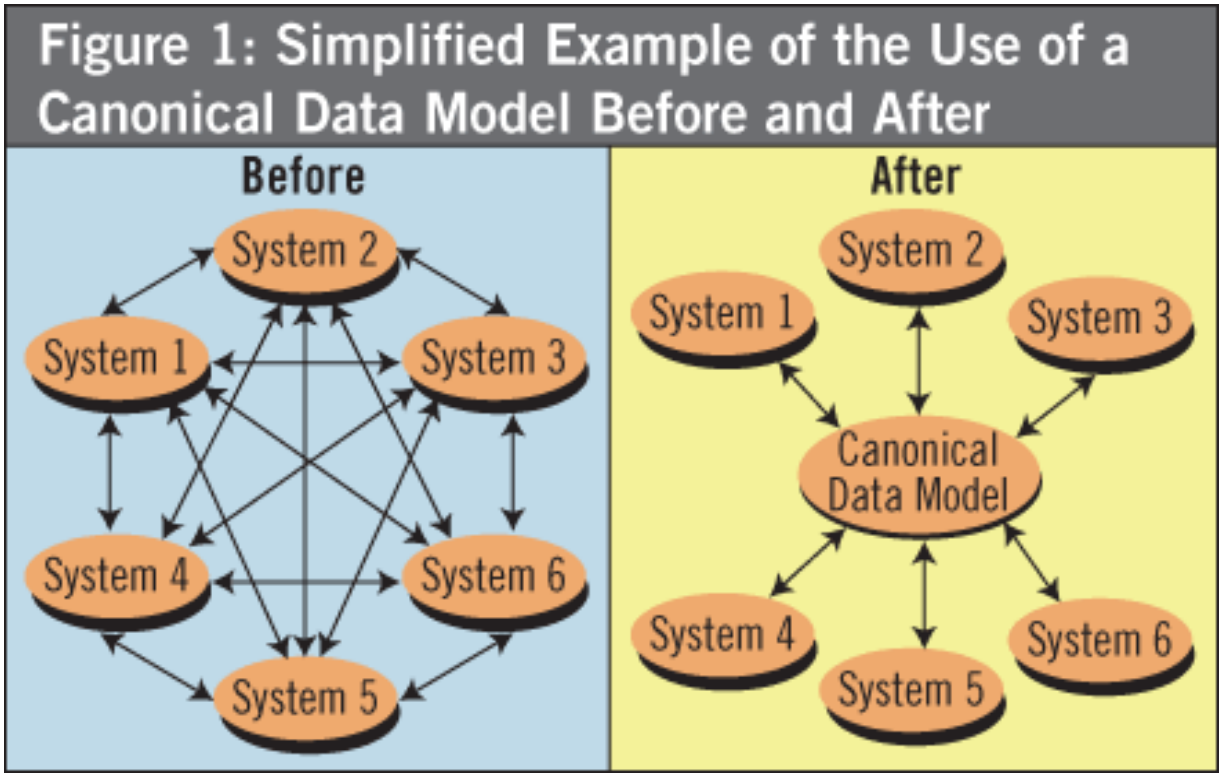
Metadata er derfor ikke bare en Nice To Have i virksomhedens arkitektur – det er et MUST. Gode metadata er grundlaget for en langsigtet, vellykket integration.

Informations Integrations Arkitekturen definerer virksomhedens Metadata uafhængig af teknologier og platforme – derfor er den brugbar i alle integrationsprojekter.

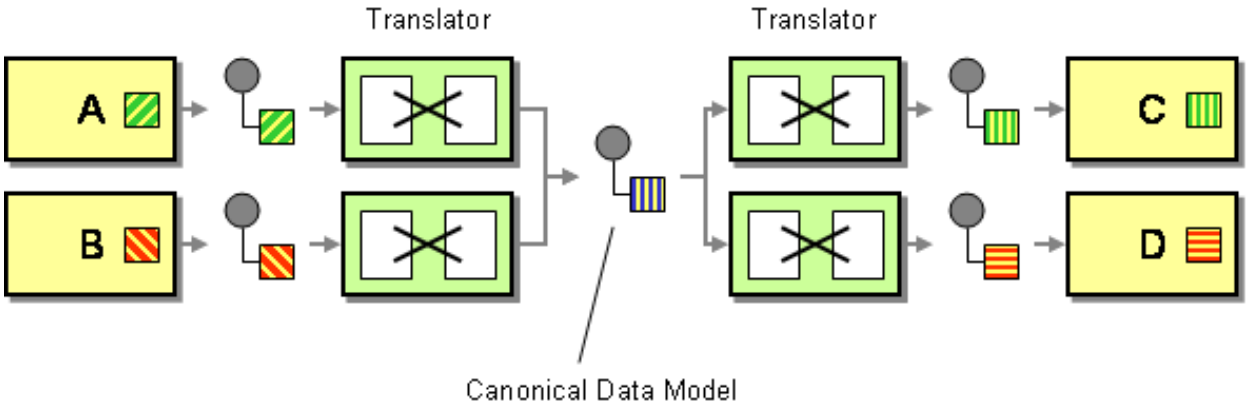
Her ses Information Integration Architecture komponentens placering i den samlede komponentarkitektur



Figuren her viser et simplificeret eksempel på en kanonisk datamodel før og efter den er implementeret



Figuren her viser fire systemer/programmer koblet til en kanonisk datamodel. Det vi ser er faktisk et Pattern for en kanonisk datamodel.



Som vi har set det tidligere, så eksisterer der også for denne komponent – Information Integration Architecture komponenten en Template.

Den ses her.

Template

• Introduktion • Scope • Deltagere • Kortlæg krav til Information Integration Design mønstre	• Data Flow Diagram • Metadata Model • Relationship Model • Review af det identificerede informationsdesign • Konklusion

Introduktion

Dette afsnit er vejledning i hvordan man skaber informationsintegrationsarkitekturen.

Scope

Scope for specifikationen for informationsarkitekturen kan være for hele virksomheden eller for et enkelt integrationsprojekt. Dokumentet skal definere behovet for informationer om forretningen, de relevante metadata og den underliggende integrationsarkitektur.

Deltagere

Dette afsnit identificerer alle interessenter i forretningen, der har kendskab til og anvender den information, der er med i integrationsprojektet - herunder virksomhedsledere, systemdesignere, arkitekter og udviklingsteamet, der skal gennemføre projektet. Alle andre deltagere og interessenter bør også identificeres - herunder deres rolle i projektet.

Kortlægge krav til Information Integration Design mønstre

Formålet med dette afsnit er at identificere og kortlægge alle de krav, der stilles til designet af informationsintegrationen.

Eksempler på områder, der er egnet til informationsintegration, er Management Dashboards; sikring af integritet i data, således at data, der ændres i et system, udbredes alle systemer, hvor de samme data optræder; fodring i realtid af systemer, der anvendes til analyse og statistik og automatisering af anvendelsen af desktop værktøjer, f.eks. Word, Excel og lignende.

Der kan naturligvis nævnes mange andre eksempler på, hvor sammenkædning og publicering af data/information giver rigtig god mening at automatisere.

Til at kortlægge disse krav, kan man anvende en Information Integration Pattern Table som vist herunder.

Application	Owner	Description	Information pattern	Source	Outcome
Management dashboard	CIO	Daily sales order volumes (world-wide)	Aggregation	Order entry systems	Graphical display of sales orders
Online customer change of address etc.	Head of sales	Update all addresses for a customer on self service change of address	Publishing	All systems containing customer addresses	Update to all these systems

Formålet her er at identificere forretningskrav hvori informations integration indgår

Eksempler på det kan være:

- Management Control Systems (dashboards) (A)
- Single views of business objects (kunde, tidsplanlægning, ….) (A)
- Data Warehouses (A)
- Systemer for undervisning (P)

hvor det gælder, at

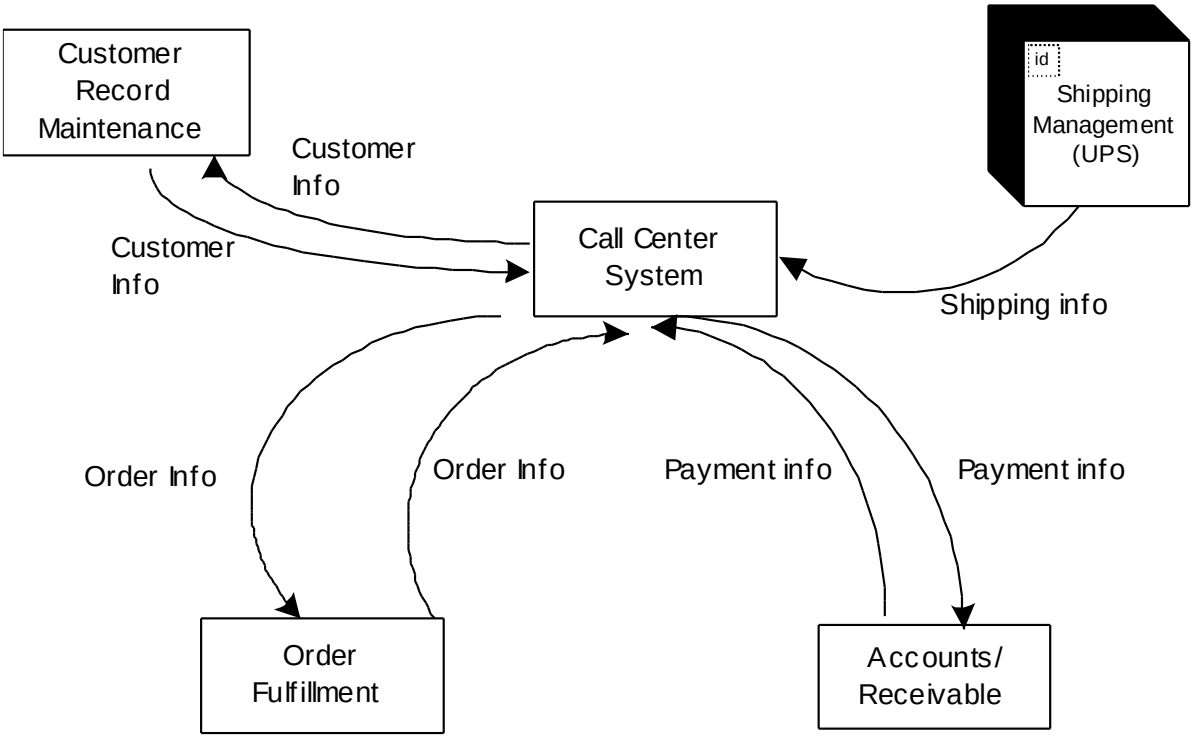
A = Aggregation
P = Publishing

Data Flow Diagram

Formålet med at skabe et Data Flow Diagram er at bestemme hvilke systemer der er involveret i data der flyder mellem systemer/programmer for senere at kunne afgøre, hvor der skal tages højde for integritetsregler i systemet.

Herunder ses et eksempel på et Data Flow Diagram for et Ordresystem. Figuren viser hvor data flyder mellem forskellige delsystemer.

Data Flow diagram



Metadata Model

En effektiv håndtering af Metqadata er essentiel for informationsdrevne arkitekturer.

Metadatamodellen bruges til at definere tilgangs- (Access) og transformationsregler.

Modellen kan også anvendes som et strategisk aktiv, der sænker omkostningerne til Operational Management og senere implementeringer. Det hjælper med at sikre datakvaliteten ved håndtering af datatilgang (Access) og integritet.

Det hjælper desuden med at maksimere den investering, der er nødvendig for at op det nødvendige systemkendskab.

Metadata Model

Basic metadata	**Data element name**	**<Student ID>**
	Data source	**<CPR register delivers data to our study adminstrative system. Alle other internal systems use study administrative system as source>**
	Description	**<In all our study administrative systems we need a unique ID for each of our students. We get it from the state CPR-register. Is used on student cards, for examination results. Whenever there is a change in a cprnr. We get a message from CPR, and our internal systems are updated every nigght>**
	Format and data type	**<DDMMYYXXXX>**
	Canonical name	**<Students ID> Name in our company – this is what it is called in all our internal systems.**
	Canonical format	**<DDMMYYXXXX> In our company**
	Transformation rules	**<DDMMYY-XXXX to DDMMYYXXXX>**
	Interface	**<Web service, adapter, API, SQL>**
Semantic metadata	Integrity rules	**<Must be unique, and a person can have only one. >**
Added security	Security parameters	**<PBA, role = Study secretary; And study secretary can only see students in own department>**
Added management	Platform	**<Hardware platform>**
	OS	**<Operating system and version>**
	DBMS	**<Database>**
	Application platform	**<Application server, other>**
	Owner of the system	**<BAAA, Study admin Office, Christian M.>**
	Location of the system	**<Directory>**
	Service information	**<Web services directory>**
	Message schema information	**<Message repository>**
	Communication protocol	**<SOAP, HTTP, TCP/IP, VAN>**
	Access mechanism	**<Enterprise message bus, message broker, JMS call, EDI VAN>**

Relationship Model

Relationsmodellen definerer afhængigheder og regler for integritet mellem dataobjekter på tværs af systemer og platforme.

Her identificeres f.eks. behov for Cascading Deletes, Rollbacks, understøttende transaktionshåndtering mm.

Her ses et eksempel på en Relationship Model

Canonical name	<Student ID>
Source system/service data element name	<What was it called where it came from? CPR-number>
Source system/service	<who brought it? Cpr register>
Business rules	<Remove dash>
Target system or service data element name	<What is called in receiving system? Student ID>
Target system or service	<Study administrative system>
Integrity rules	<Rollback and compensation rules – cascading deletes and other ripple effects>
Security requirements	<Encryption, nonrepudiation, and access rules>

Review af det identificerede informationsdesign

Review, godkendelse og accept er kritisk for succesen af det endelig resultat og den fremtidige agilitet af systemet.

Alle relevante interessenter – minimum dem som er identificeret i afsnittet Deltagere – skal deltage i dette Review, verificere og godkende modellen.

Følgende kan være guidelines for succesfuld Review:

- Vær sikker på, at alle interessenter er repræsenteret
- Forklar processen og grundlæggende regelsæt før Review
- Kritisér designet – gå ikke efter personer
- Designerne skal forklare og argumentere for trufne valg – gå ikke i forsvarsposition
- Identificere 'ejere' af data/information
- Identificere de systemer hvor data/information 'fødes'
- Definér hvad der forstås ved kvalitet af data/informationen

Konklusion

Her sammenfattes de endelige kommentarer til designet og brugen af systemet.

Det anbefales at oprette et Metadata Repository og ikke mindst er det vigtigt at sikre en løbende vedligeholdelse af denne

Endelig er det vigtigt at inkludere semantisk betydning i metadata – det er ikke altid så nemt.