Generel systemmodellering og systemudvikling af computerbaserede systemer

Formulering af et generelt systemmodelleringssprog som fundament til forbedring af generelle, grundlæggende problemer i forbindelse med systemudvikling af computerbaserede systemer

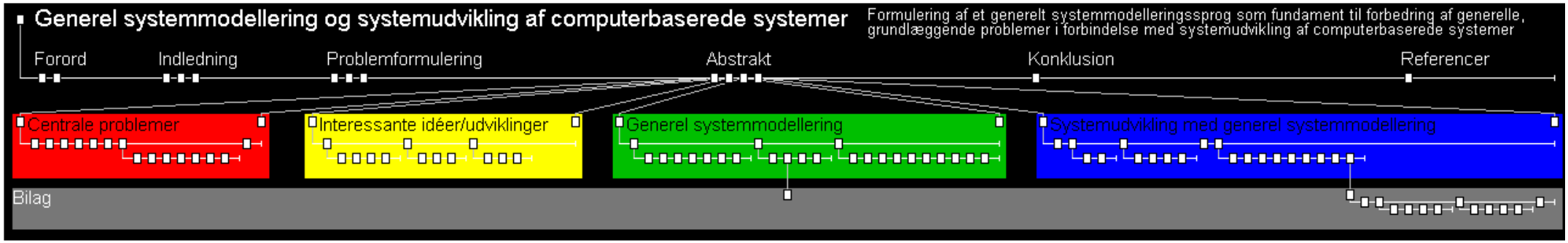
Søren Vejrum

kandidatafhandling

cand.merc.dat.

Handelshøjskolen i København

18. juni 1999



Referencer

- 3D Visualization of Software Architectures
- Advanced Programming Language Design
- Against Method
- Algorithmic Graph Theory
- Algorithms & Data Structures = Programs
- Algorithms for Graph Partitioning: A Survey
- Anti Patterns
- Analysis Patterns
- Art of Computer Programming, Vol. 1: Fundamental Algorithms
- Automated Consistency Checking of Requirements Specifications
- Automated Tool for Analyzing Completeness of Equational Specifications
- Brief Introduction to Coloured Petri Nets
- CDIF Framework for Modeling and Extensibility
- Completeness and Consistency Analysis of State-Based Requirements
- Computers in Context
- Connecting the Design of Software to the Design of Work
- Consumer Spectrum, The
- Coordination Languages and their Significance
- Coordination Models and Languages as Software Integrators
- CORBA
- CORBA Component Scripting
- Critical Issues in Systems Theory and Practice
- Critical Systems Thinking
- Design as Practiced
- Design for People at Work
- Design Languages
- Design Methodology Management Using Graph Grammars
- Design Patterns
- DEVS Formalism and Methodology
- Does 'Information Systems' Need Systems?
- End-User Construction of Distributed Multimeddia Applications
- Facets of Systems Science
- Flow-Based Programming
- Footholds for Design
- Formal Methods: State of the Art and Future Directions
- Formal Semantics of Data Flow Diagrams
- Formal Transformations
- Four Dark Corners of Requirements Engineering
- Frame-Based Software Engineering
- From Object-Oriented to Goal-Oriented Requirements Analysis
- General Principles of Systems Design
- General System Theory
- General Systems Theory - The Skeleton of Science
- Graph Drawing: Algorithms for the Visualization of Graphs
- Guide to the SQL Standard
- Handbook of Graph Grammars and Computing by Graph Transformation: Foundations, Vol. 1
- High Level Language for Specifying Graph Based Languages and their Programming Environments
- Informal and Formal Requirements Specification Languages
- Information System Behavior Specification by High-Level Petri Nets
- Information Systems Development
- Introduction to General Systems Thinking
- Introduction to the Practical Use of Coloured Petri Nets
- Introduction to the Theoretical Aspects of Coloured Petri Nets
- Keeping It Simple
- Lightweight Approach to Formal Methods
- Lucid Primer
- M68000 8-/16-/32-Bit Microprocessors Reference Manual
- Managing the Development of Large Software Systems
- Meta Object Facility
- Meta-Modeling and Modularity
- Modern Compiler Implementation
- Modern Operating Systems
- Modern Structured Analysis
- Multiview
- Mythical Man-Month, The
- No Silver Bullet
- "No Silver Bullet" Refired
- Object-Oriented Concepts, Databases, and Applications
- Object-Oriented System Development
- On Visual Formalisms
- Operating Systems
- Process Modeling
- Professionel Systemudvikling
- Program Visualization: the Art of Mapping Programs to Pictures
- Programmers' Playground, The
- Projektledelse i løst koblede systemer
- Rational Design Process, A
- Reflective Conversation with Materials
- Requirements Tracing
- Rethinking Systems Analysis & Design
- Semantically Extended Data Flow Diagrams: A Formal Specification Tool
- Separating Fact from Fiction in Software Architecture
- sNet Formalism - Technical Report
- Social Network Analysis: Methods and Applications
- Social Structures: A Network Approach
- Soft Systems Methodology in Action
- Software Architecture Characterization
- Software Architecture Recovery and Restructuring through Clustering Techniques
- Software Architecture Styles as Graph Grammars
- Software Development Environment
- Software Engineering
- Software Engineering Environments
- Software Engineering with Reusable Designs and Code
- Software Requirements & Specifications
- Software-ICs
- Spiral Model of Software Development and Enhancement
- Strengthening the Bridge between Qualitative and Quantitative Modeling
- Structure in Fives
- Structured Computer Organization
- SVDM: An integrated combination of SA and VDM
- System Development
- Systems: Concepts, Methodologies and Applications
- Theory of Modelling and Simulation
- Tools & Materials Approach for Large Scale OO Software Development
- Toward Formalizing Structured Analysis
- Understanding and Deploying LDAP Directory Services
- Unified Modelling Language
- Using Style to Understand Descriptions of Software Architecture
- Visualizing and Querying Software Structures
- Who's Afraid of Ontologies?

Forord og læsevejledning

Denne rapport er udarbejdet som kandidatafhandling på cand. merc. dat. studiet på Handelshøjskolen i København. Rapporten er udarbejdet af Søren Vejrum med Leif Bloch Rasmussen som vejleder.

_____

Søren Vejrum, 18. juni 1999

Rapportudgaver

Rapporten er udarbejdet som et hypertekst dokument i form af et såkaldt "trelligram" samt som World Wide Web sider. Denne udgave af rapporten inkluderer digitale kopier af benyttet reference litteratur som bilag i det omfang disse har været tilgængelige. NB: Disse bilag er kun til personlig, uddannelsesmæssig brug og kopieringsrettigheder er ikke indhentet fra ophavsretshaverne. Derudover er rapporten udarbejdet i en papirudgave uden reference litteratur som bilag.

Trelligram udgaven af rapporten kræver Microsoft Windows 95/98/NT styresystem for at kunne læses. World Wide Web udgaven af rapporten kræver installation af World Wide Web browser software for at kunne læses. Reference litteratur er vedlagt som bilag i form af World Wide Web sider eller Adobe Acrobat PDF filer og kræver installation af World Wide Web browser og Adobe Acrobat Reader for at kunne læses.

Læsevejledning

Rapporten er udformet som et antal sidesekvenser af forskellig detaljeringsgrad. Den overordnede sekvens opsummerer alle væsentlige problemstillinger og konklusioner og kan læses separat. Derudover er de fire hovedtemaers problemstillinger og konklusioner hver især uddybet i separate sekvenser med mere detaljeret argumentation og litteratur referencer. Det anbefales at rapportens overordnede sekvens læses først i sin helhed, hvorefter de detaljerende sekvenser kan læses.

Da rapporten er udarbejdet som et hypertekst dokument er de enkelte sider og sidegrupperinger formuleret således at de i et vist omfang kan læses som enkeltstående, uafhængige afsnit. Dette medfører dog også en vis grad af overlapninger og gentagelser ved sekventiel læsning, som eventuelt kræver lidt tilvænning for læseren i forhold til sekventielt formulerede rapporter.

De enkelte sider er opbygget med et grafisk oversigtskort for rapportens struktur, sekvenser og sider øverst på siden. Oversigtskortet er "aktivt" med mulighed for hurtig bladring til de enkelte sider. Oversigtskortet findes dog ikke i papirudgaven af rapporten. I venstre kolonne på siden findes et antal navigeringspile til bladring mellem rapportens sider - frem og tilbage i siderne for den aktuelle sekvens samt op og ned for skift mellem de forskellige detaljeringsniveauer. Umiddelbart under navigeringspilene findes en liste over detaljerende sider (hvis nogen), og derunder en liste over litteratur referencer (hvis nogen) for den aktuelle side. De listede detaljerende sider og litteratur referencer er "aktive" for umiddelbar bladring til de pågældende sider. Papirudgaven er organiseret i "naturlig" rækkefølge til sekventiel læsning af siderne.

Af hensyn til den umiddelbare læsevenlighed af teksten er litteratur referencer ikke angivet i selve teksten men i venstre margin medmindre at det drejer sig om direkte citater eller henvisninger. Ved angivelse af litteratur referencer i venstre margin er teksten i det pågældende afsnit delvist en syntese og/eller omtale af idéer præsenteret i litteratur referencerne samt egne idéer og udsagn.

Det antages at læseren har et bredt kendskab til computerbaserede systemer og systemudvikling, og "almindelige" teorier, metoder og begreber introduceres derfor ikke nærmere i rapporten. For yderligere information henvises til den refererede litteratur med mere detaljerede beskrivelser af de anvendte teorier, metoder og begreber.

Indholdsfortegnelse

prev next

Generel systemmodellering og systemudvikling af computerbaserede systemer

Del A: Centrale problemer ved systemudvikling

Del B: Interessante idéer og udviklinger indenfor systemudvikling

Del C: Generel systemmodellering

Del D: Systemudvikling med generel systemmodellering

Bilag

Kompleksitet

prev next

Brooks [1986] No Silver Bullet

Christensen/Kreiner [1994] Projektledelse i løst koblede systemer

Hvorfor skal det være så svært at udvikle computerbaserede systemer? Skal det være så svært at udvikle computerbaserede systemer? Dette er spørgsmål, som jeg ofte har stillet mig selv, og som mange andre også har stillet sig selv og andre.

En stor del af besværet skyldes uundgåelig kompleksitet ved problemerne, løsningerne og processerne i forbindelse med systemudvikling. Ofte er målet med systemudvikling ikke blot et spørgsmål om at implementere eksisterende viden i et computerbaseret system, men derimod et forsøg på at løse, forbedre eller i hvert fald forandre problemer på grænsen af eller udover vores eksisterende viden. Problemerne indgår desuden ofte i uberegnelige/uforudsigelige sociale og organisatoriske systemer. Løsningerne er sjældent rene standardløsninger, som uden videre kan anvendes i den enkelte problemsituation. For det meste involverer løsningen i et vist omfang nyudvikling af helt nye løsninger eller videreudvikling af eksisterende løsninger. Løsningerne er ofte teknisk meget omfattende og komplekse, og computerbaserede systemer er af natur i vid udstrækning usynlige og uhåndgribelige abstraktioner. Med delvist ukendte problemer og delvist ukendte løsninger følger nødvendigvis også delvist ukendte udviklingsprocesser, og usikkerheden øges af eventuelle forandringer i problemerne og løsningerne i løbet af udviklingstiden.

Dermed bliver det sikkert ved med at være svært at udvikle computerbaserede systemer, da computeren er vores kraftigste og derfor foretrukne værktøj til at løse mange svære problemer. Systemudvikling vil nærmest per definition altid omhandle nye, ukendte løsninger, da rent kendte løsninger ikke genudvikles men blot kopieres. Usikkerheden kan heller ikke undgåes, da der er tale om to forskellige, modsat rettede former for usikkerhed: kravene til systemet og dermed systemudviklingsprocessen ved specifikationen kontra systemets anvendelighed ved ibrugtagningen.

En stor del af besværet skyldes efter min mening dog også unødvendig kompleksitet som følge af de værktøjer og metoder, som vi idag anvender til systemudvikling. Metoderne er ofte for simple i forhold til alle variationerne i problemerne og systemerne, som forsøges håndteret med metoderne, således at 'virkeligheden' må tilpasses metoderne, eller at systemudviklerne må opfinde ændringer/tilføjelser til metoderne. Omvendt er metoderne også ofte for komplicerede/omfattende i forhold til de aktuelle problemer og behov i forbindelse med det enkelte system, således at de medfører unødvendigt arbejde. Værktøjerne er ofte lavet snævert til bestemte metoder og med begrænset fleksibilitet, således at problemerne ved metoderne forstærkes kraftigt. Værktøjerne er desuden ofte komplicerede at lære og at bruge, og forskellige værktøjer er svære at få til at arbejde sammen. Metoderne og værktøjerne er som regel dårligt egnede til ukendte og foranderlige systemudviklingsprocesser. Dette betyder at systemudvikling forhåbentligt kan blive nemmere, hvis (en del af) denne unødvendige kompleksitet fjernes.

Udvikling i systemudvikling

prev next

Brooks [1986] No Silver Bullet

Brooks [1995] "No Silver Bullet" Refired

Det er med meget blandede følelser, at man følger med i udviklingen indenfor teori og praksis for systemudvikling. Der bliver jævnligt lovet revolutionerende fremskridt fra såvel teoretikere som praktikere ved hjælp af en ny metode eller et nyt værktøj. Der sker da også forbedringer, men som regel er der langtfra indfrielse af forventningerne. Løfterne om forbedringer ligger langt over de reelle/oplevede forbedringer. Manglende indfrielse af urealistiske "silver bullet" løfter/forventninger er ikke så overraskende, men ofte indfries selv moderate, realistiske løfter/forventninger heller ikke.

Der sker reelle forbedringer, men de er ofte små og isoleret til enkelte dele af systemudviklingen, således at de opvejes af nye direkte eller indirekte problemer på tilstødende områder som følge af forbedringerne. Det er de færreste metoder og værktøjer, som umiddelbart kan indpasses i systemudviklingen. De eksisterende metoder og værktøjer skal eventuelt helt kasseres, eller der skal laves nye grænseflader mellem de forskellige metoder og værktøjer i form af ressource- og tidskrævende manuelle procedurer, automatiske oversættere eller integrerede tilpasninger. Skift af metoder og værktøjer i systemudvikling har desværre større lighed med religionsskifte end en håndværkers udskiftning af/supplering med nye metoder og værktøjer.

På trods af den store indsats for at forbedre systemudvikling opnåes der kun begrænsede resultater i praksis, og man genoplever/fortsætter ofte med de gamle, velkendte, store problemer. Når dette er tilfældet (eller i hvert fald oplevelsen) er det nok værd at overveje om der er noget grundlæggende galt med den måde hvorpå vi udvikler computerbaserede systemer, således at de mange forbedringsinitiativer kun bliver lappeløsninger og skaber/bevarer unødvendige problemer for sig selv.

Udsynsfelt

prev next

Ashcroft & Wadge [1995] Lucid Primer

Brooks [1986] No Silver Bullet

Norman [1996] Design as Practiced

Der bliver brugt mange ressourcer på at forbedre systemudvikling gennem udvikling af nye varianter af velkendte metoder og værktøjer. Der findes efterhånden et nærmest uendeligt og stadigt voksende antal minimalt forskellige programmeringssprog, modellerings-/diagrammeringsteknikker og analyse/design metoder. Der synes at være en ukuelig tro på, at man bare lige skal gøre det lidt anderledes, så bliver det meget bedre. Grundlæggende ændrer det dog som regel ikke meget og det er vanskeligt at berettige store omkostninger ved at skifte programmeringssprog, modellerings-/diagrammeringsteknikker og analyse/design metoder.

I stedet for til stadighed at forsøge at forbedre lidt på de velkendte værktøjer og metoder, må der være et væsentligt større potentiale for forbedring af systemudvikling ved at kigge bredere og dybere for grundlæggende problemer og løsninger. Dette har tidligere resulteret i væsentlige forbedringer specielt ved overgangen fra maskinekode til højniveau programmeringssprog og integrerede programmeringsmiljøer, og der kan eventuelt/formodentligt stadig opnåes væsentlige forbedringer gennem grundlæggende ændringer.

Fagområdet 'generel systemteori' er specielt kendetegnet ved at kigge bredt og dybt gennem analyse af forskellige problemer og teorier på tværs af forskellige fagområder med henblik på at identificere og generalisere tværgående, fælles/relaterede systemiske egenskaber og teorier.

Problemformulering

prev next

Centrale problemer ved systemudvikling

Interessante idéer og udviklinger indenfor systemudvikling

Generel systemmodellering

Systemudvikling med generel systemmodellering

Målet med denne rapport er med udgangspunkt i generel systemteori at foretage en bred analyse af det samlede systemudviklingsforløb for at identificere og generalisere nogle tværgående, fælles/relaterede problemer og udviklingstendenser samt at formulere et forslag til afhjælpning af disse problemer gennem:

- en bred analyse af centrale problemer i systemudvikling med henblik på identificering af grundlæggende, generelle problemer

- analyse og generalisering af interessante, "nye" udviklinger/idéer indenfor systemudvikling

- formulering af et generelt systemmodelleringssprog til systemudvikling

- beskrivelse af systemudvikling med det formulerede generelle systemmodelleringssprog og generelle systemmodelleringsværktøjer

Afgrænsning

prev next

Centrale problemer ved systemudvikling

Interessante idéer og udviklinger indenfor systemudvikling

Generel systemmodellering

Systemudvikling med generel systemmodellering

Målet med analysen af centrale problemer i systemudvikling er at foretage en bred analyse af udvalgte, kendte problemer baseret på eksisterende kilder. Analysen forsøges på ingen måde gjort udtømmende bortset fra at den skal dække hele systemudviklingsforløbet. De enkelte problemer analyseres desuden ikke detaljeret men refereres blot til eksisterende kilder.

Målet med analysen af interessante, "nye" udviklinger/idéer indenfor systemudvikling er ligeledes at foretage en ikke udtømmende, bred analyse af udvalgte udviklinger/idéer i relation til de identificerede centrale problemer og formulering af generel systemmodellering i forbindelse med systemudvikling. De enkelte udviklinger/idéer analyseres ikke detaljeret men refereres blot til eksisterende kilder.

Formuleringen af et generelt systemmodelleringssprog til systemudvikling laves således at den er realistisk og anvendelig, men det sikres dog ikke at den er garanteret anvendelig til enhver situation, og den generelle systemmodellering valideres ikke matematisk eller på tilsvarende måde.

Beskrivelsen af systemudvikling med den formulerede generelle systemmodellering omfatter ikke et komplet systemudviklingsforløb af et komplet system men kun udvalgte dele af et systemudviklingsforløb og et system som eksempel på hvordan det eventuelt kan gøres.

Definitioner

prev next

"Generel systemudvikling" og "generel systemmodellering" anvendes her som betegnelse for alle systemudviklingsaktiviteter og systemmodelleringsaktiviteter i forbindelse med et komplet systemudviklingsforløb for systemudvikling af et komplet system. Generel systemudvikling og systemmodellering bør principielt dække systemudvikling og systemmodellering af alle mulige systemer, men her fokuseres der dog primært på systemudvikling og systemmodellering i forbindelse med (delvist) computerbaserede systemer.

"Traditionel systemudvikling" og "traditionel systemmodellering" anvendes her som betegnelse for eksisterende systemudvikling og systemmodellering som modstykke til generel systemudvikling og systemmodellering. Traditionel systemudvikling og systemmodellering omfatter her således også moderne systemudvikling og systemmodellering som for eksempel objektorienteret systemudvikling og systemmodellering.

"Computerbaseret system" omfatter her det tekniske computer system i form af computer hardware og software samt eventuelt relevante omkringliggende systemer i form af mennesker (som brugere og organisationer) og maskiner (herunder andre computer systemer), samt disses indbyrdes relationer og relationer til det tekniske computer system.

Abstrakt af centrale problemer ved systemudvikling

prev next

Centrale problemer ved systemudvikling

Den centrale aktivitet ved systemudvikling af computerbaserede systemer er systemmodellering. Dels systemmodellering af "virkeligheden" i form af systemets domæne og omgivelser i bred forstand, den aktuelle problemsituation der har givet anledning til systemudviklingsforløbet samt de fænomener systemet omfatter med henblik på at opnå forståelse af systemet som udgangspunkt for systemudviklingen. Dels systemmodellering af et computerbaseret system og dets indplacering i omgivelserne med henblik på forbedring af problemsituationen.

Systemmodellerne og systemodelleringen varierer således meget fra systemmodellering som fortolkning og simplificering af vilkårlige fysiske og sociale fænomener til systemmodellering som konstruktion af selvstændige, formelle, logiske maskiner implementeret i en bestemt computertype.

Traditionel systemudvikling omfatter mange forskellige, separate systemmodeller i forbindelse med de forskellige systemudviklingsaktiviteter samt mange forskellige systemmodelleringssprog og tilhørende systemmodelleringsmetoder og -værktøjer som systemudviklerne skal beherske.

Systemmodellering med forskellige, separate systemmodeller er anvendelig ved simple, ensrettede, fremadskridende systemudviklingsforløb som den klassiske vandfaldsmodel og iterative videreudviklinger, hvor systemmodeller kan "oversættes" fra den ene systemudviklingsaktivitet til den næste. Forskellige, separate systemmodeller er dog uhensigtsmæssige ved mere dynamiske og eventuelt kaotiske, eksperimenterende systemudviklingsforløb med vilkårlige skift mellem de forskellige systemudviklingsaktiviteter, da det er nødvendigt med dobbeltrettede "oversættelser" mellem alle kombinationer af systemmodeller når ændringer af en systemmodel kræver tilsvarende ændringer af de andre systemmodeller for at sikre konsistens mellem dem.

Der findes og anvendes desuden et stort antal forskellige systemudviklingsmetodologier med forskellige systemmodeller og varianter af systemmodeller samt tilknyttede systemmodelleringsværktøjer uden umiddelbar mulighed for udskifte/kombinere dem. Det betyder store begrænsninger for systemudviklernes valgfrihed mellem forskellige systemudviklingsmetodologier, -metoder og -værktøjer og medfører store anskaffelses- og uddannelsesomkostninger for systemudviklerne ved udskiftning af systemudviklingsmetodologier, -metoder og -værktøjer.

Endeligt er systemmodellerne og systemmodelleringsværktøjerne bundet til bestemte systemmodelelementer uden mulighed for at supplere med yderligere eller udskifte med andre systemmodelelementer efter behov. Det betyder at systemudviklerne eventuelt må se bort fra relevante aspekter og begrænse systemmodelleringen til de af systemmodelleringsværktøjerne understøttede systemmodelelementer eller må supplere systemmodelleringen med ad hoc systemmodeller ved hjælp af generelle værktøjer (som for eksempel tekstbehandling- og tegneprogrammer).

Der er således behov for integreret systemmodellering, der understøtter dynamiske systemudviklingsforløb, samt fleksible systemmodeller og systemmodelleringsværktøjer, der muliggør tilpasning af systemmodellerne efter individuelt behov for den enkelte systemudvikler og det enkelte systemudviklingsforløb og muliggør fri kombination og udskiftning af systemmodelleringsværktøjer.

Abstrakt af interessante idéer og udviklinger indenfor systemmodellering

prev next down

Interessante idéer og udviklinger indenfor systemudvikling

Der er til stadighed mange tiltag til forbedring af systemudvikling og systemmodellering, og selvom de er meget forskellige og fragmenterede, kan der ses nogle generelle udviklingstendenser

Systemmodellering bliver i stigende grad forenet igennem hele systemudviklingsforløbet med udbredelsen af objektorienteret systemudvikling og anvendelse af fælles objektorienterede koncepter for systemanalyse, systemdesign, programmering, og system-/softwarearkitekturer samt standardiserede systemmodelleringssprog for systemanalyse og systemdesign. Dette begrænser antallet af koncepter og systemmodelleringssprog som systemudviklerne skal beherske, og det øger systemudviklernes valgfrihed mellem systemmodelleringsværktøjer. Objektorienteret systemmodellering omfatter dog fortsat forskellige systemmodeller for systemanalyse, systemdesign og programmering samt forskellige systemmodelleringssprog og systemmodelleringsværktøjer for systemanalyse/-design og programmering.

Der er ligeledes i stigende grad forenet systemmodellering samt formalisering af systemmodellering med integration af formelle systemmodeller og uformelle systemmodeller samt videreudvikling af simple, generelle systemmodeller, systemmodelleringssprog og -værktøjer som Petri Net fra at være orienterede mod simulering af mindre, tekniske systemer med simple data og processer til også at være egnede til systemmodellering af større, generelle systemer med komplekse data og processer. Dette danner udgangspunkt for øget kvalitet af systemmodeller (og herunder computerbaserede systemer) igennem matematisk/logisk verifikation af; mere avancerede systemmodelleringsværktøjer i form af matematisk/logisk analyse og bearbejdning; samt øget automatisering af systemmodellering med algoritmisk transformation af formelle systemmodeller.

Derudover er der øget formalisering af systemmodellering med formulering af sproglige begreber og definitioner for mønstre af tilbagevendende systemmodelkonstruktioner. Med dette er der udgangspunkt for mere effektiv systemmodellering igennem øget genbrug samt bedre kommunikation mellem systemudviklere omkring systemmodeller

Den store udbredelse af objektorientering kan ses som blot ét skridt i retning af øget flow orientering i forbindelse med computerbaserede systemer med systemer struktureret som selvstændige, kommunikerende delsystemer. Distribuerede computerbaserede systemer sikrer kapacitet og stabilitet med fysisk adskilte computere/processorer der kommunikerer over netværk/databusser. Objektorienteret programmering strukturerer programmer som selvstændige objekter der sender beskeder med data til hinanden, og floworienterede koordineringssprog og programmeringssprog rendyrker dette yderligere med netværk af enkelte programinstruktioner, der er forbundet igennem udveksling af data. Endeligt kan computerbaserede systemer i stigende grad konfigureres af slutbrugerne, hvilket mest fleksibelt sker med konstruktion af kommunikationsnetværk over hvilke separate applikationskomponenter udveksler data.

Der kan således ses udviklingstendenser i retning af mere generelle systemmodeller, systemmodelleringssprog og systemmodelleringsværktøjer; mere formaliserede systemmodeller og systemmodelleringsmetoder uden begrænsninger til matematiske systemmodeller og simple systemer; samt floworientering som hensigtsmæssig og eventuelt nødvendig systemopfattelse for systemmodellering og systemkonstruktion af store og komplekse systemer.

Disse udviklingstendenser bør inddrages i overvejelser om og kan eventuelt anvendes som udgangspunkt for en eventuel reformulering af systemmodellering og formulering af et nyt systemmodelleringssprog.

Abstrakt af generel systemmodellering

prev next down

Generel systemmodellering

Generel systemteori omfatter ikke umiddelbart færdigtformulerede detaljerede og integrerede systemparadigmer, systemmodeller og systemmodelleringssprog der kan anvendes til systemudvikling og systemmodelllering af computerbaserede systemer. Der er dog formuleret forskellige grundlæggende teorier for systemer og systemmodellering, som også er relevante i forbindelse med systemmodellering af computerbaserede systemer.

Med generel systemteoris fokus på systemer i form af organiseret kompleksitet som en tværfaglig disciplin for såvel hårde, maskinelle systemer som bløde, sociale systemer, omfatter det i høj grad også computerbaserede systemer i form af logiske maskiner samt deres indplacering i og interaktion med omgivelserne. Generelle systemteorier for systemkoncepter, systemers bestanddele og strukturer, systemiske egenskaber, systemtyper, systemmodellering og håndtering af systemkompleksitet gælder således også for computerbaserede systemer. Derudover tilbyder relaterede matematiske discipliner generelle metoder til analyse og bearbejdning af generelle systemmodeller.

Generel systemteori er således også allerede indraget i forbindelse med systemudvikling af computerbaserede systemer, og generel systemteori videreudvikles desuden i vid udstrækning med udgangspunkt i computerbaserede systemer.

Med udgangspunkt i generel systemteori og matematiske grafer formuleres der dermed et generelt systemmodelleringssprog til anvendelse til systemudvikling af computerbaserede systemer. Det formulerede, generelle systemmodelleringssprog omfatter udelukkende nogle få, generelle systemmodelleringskoncepter, og det er formuleret med henblik på, at det principielt skal kunne anvendes til integreret systemmodellering for alle systemudviklingsaktiviteter i forbindelse med et systemudviklingsforløb.

Der er desuden defineret en formel semantik for det formulerede, generelle systemmodelleringssprog, og systemmodelleringssproget omfatter mulighed for men ikke krav om formalisering af systemmodeller i form af systemmodelleringskoncepter til klassificering af delsystemer/systemelementer.

Endeligt er det illustreret hvordan det formulerede, generelle systemmodelleringssprog kan anvendes til systemmodellering af såvel statiske datastrukturer som dynamiske processtrukturer i form af floworienterede netværk for computerbaserede systemer.

Abstrakt af systemudvikling med generel systemmodellering

prev next down

Systemudvikling med generel systemmodellering

Det formulerede, generelle systemmodelleringssprog omfatter udelukkende semantisk definition for de få, simple, generelle systemmodelleringskoncepter. For at dette er praktisk anvendeligt til systemudvikling af computerbaserede systemer skal det understøttes af generelle systemmodelleringsværktøjer, der supplerer semantikken med standard repræsentation af generelle systemmodeller i computerbaserede formater; forskellige generelle præsentationsformer af generelle systemmodeller for systemudviklere; samt forskellige generelle analyseværktøjer og definitioner for eksisterende, specielle analyseværktøjer.

Lagring af generelle systemmodeller bør ske i et antal standardformater for henholdsvis computerbaserede filer og databaser, således at systemudviklere og forskellige, generelle systemmodelleringsværktøjer umiddelbart kan udveksle generelle systemmodeller. Sådanne standardformater for lagring af generelle systemmodeller kan bestå af eller baseres på de forskellige, eksisterende systemmodel formater.

Systemudvikling af computerbaserede systemer med generel systemmodellering omfatter systemmodellering af mange forskellige fænomener fra uformelle, sociale systemmodeller til formelle, maskinelle systemmodeller som alle beskrives med de få, simple, generelle systemmodelleringskoncepter. De er således alle modelleret som matematisk graf baserede netværk og hierarkier, men disse kan præsenteres på forskellige måder, der fremhæver forskellige aspekter ved de generelle systemmodeller. Et generelt systemmodelleringsværktøj bør dermed omfatte forskellige præsentationsformer, som systemudviklerne frit kan vælge og skifte mellem til at visualisere de generelle systemmodeller i form af for eksempel verbale, matematiske/logiske samt forskellige to- og tre-dimensionelle grafiske formater.

Et generelt systemmodelleringssprog bør desuden omfatte mulighed for at definere forskellige udsnit af eventuelt meget omfattende og komplekse, generelle systemmodeller, som kun omfatter de relevante delsystemer for de enkelte systemudviklingsaktiviteter og systemudviklere. Sådanne udsnit kan formuleres med en form for forespørgselssprog, der kan specificere simple udsnit og eventuelt komplekse transformationer af generelle systemmodeller.

Generelle systemmodelleringsværktøjer bør ligeledes omfatte forskellige, generelle analyseværktøjer. Sådanne generelle analyseværktøjer baseret på matematisk graf baserede metoder kan anvendes til at analysere generelle systemmodeller for forskellige karakteristika og validering udfra givne kriterier samt til forskellige omstruktureringer og transformationer af generelle systemmodeller.

De mange forskellige systemmodelleringsaktiviteter i forbindelse med systemudviklingsforløb for systemudvikling af computerbaserede system vil alle kunne ske ved hjælp af sådanne generelle systemmodelleringsværktøjer med ét generelt systemmodelleringssprog og mulighed for én integreret systemmodel for det samlede system.

Systemudviklingsforløbet er dermed ikke bundet til givne systemudviklingsaktiviteter og givne relationer mellem disse af hensyn til systemmodelleringsværktøjerne og systemmodelleringen men kan derimod frit sammensættes og ændres udfra de aktuelle behov for det enkelte systemudviklingsforløb og de enkelte systemudviklere. Der er principielt ingen forskel på de forskellige systemudviklingsaktiviteter i forhold til selve systemmodelleringen. Forskellene ligger udelukkende i forskelligt fokus på dele af og aspekter ved systemet samt i detaljerings- og formaliseringsgraden af systemmodellerne.

Konklusion

prev next

Der er her formuleret en semantik for et generelt systemmodelleringssprog, og det er illustreret hvordan en række forskellige, repræsentative systemmodelleringsaktiviteter igennem et komplet systemudviklingsforløb af et computerbaseret system kan ske ved hjælp af det formulerede generelle systemmodelleringssprog.

Det foreslåes således at generel systemmodellering kan danne grundlag for forbedring af systemudvikling af computerbaserede systemer ved at fjerne eller i hvert fald mindske to centrale problemer i forbindelse med sådanne systemudviklingsforløb. Generel systemmodellering danner grundlag for øget integration i systemmodelleringen igennem hele det komplette systemudviklingsforløb med mulighed for opbygning af én integreret systemmodel. Generel systemmodellering danner ligeledes grundlag for øget fleksibilitet for systemudviklingsforløb med mulighed for vilkårlig og eventuel ad hoc definition af forskellige systemudviklingsaktiviteter og systemmodel(ler).

For praktisk anvendelighed af generel systemmodellering er der dog behov for implementering af generelle systemmodelleringsværktøjer med generelle præsentationsformer og analysemetoder. Derudover er der specielt også behov for opbygning af et bibliotek af standard moduler med generelle systemmodel definitioner for eksisterende computer systemarkitekturers processorer og styresystemer samt programmeringssprog og andre systemudviklingsværktøjer således at generel systemmodellering kan anvendes ligeså effektivt som og eventuelt kombineres med eksisterende systemudviklingsmetoder.

Systemudvikling af computerbaserede systemer med generel systemmodellering kan eventuelt umiddelbart virke meget anderledes and traditionel systemudvikling, men det kan dog også blot ses som en videreføring og generalisering af eksisterende idéer og udviklingstendenser.

Objektorienteret systemudvikling introducerer fælles systemmodelleringskoncepter og -sprog i forskellige dele af systemudviklingsforløbet, og generel systemmodellering viderefører dette til alle dele af systemudviklingsforløbet. Objektorienteret systemudvikling introducerer ligeledes opdeling af programmer i principielt selvstændige, kommunikerende delsystemer i form af objekter til bedre håndtering af kompleksitet og programgenbrug, og generel systemmodellering går videre med principiel opdeling i separate delsystemer helt ned til de enkelte lavniveau programinstruktioner og dataelementer i form af floworientering som gennemgående koncept (uden at dette nødvendigvis også forudsætter floworienteret implementering med anvendelse af floworienterede programmeringssprog og maskinarkitekturer med mere).

Generel systemmodellering omfatter desuden et formelt semantisk grundlag, der kan danne udgangspunkt for bredere anvendelse af formelle metoder til analyse og bearbejdning af systemmodeller samt eventuelt for integration med for eksempel Petri Net og DEVS med henblik på analyse og simulering af systemmodeller. Generel systemmodellerings semantik stiller dog ikke omfattende krav til formalisering og er umiddelbart primært orienteret mod forholdsvis simpel sikring af entydighed, konsistens og komplethed igennem anvendelse af type definitioner. Der kan således formodentligt være stort udbytte ved integrering af generel systemmodellering med eksisterende formelle systemmodellerings- og simuleringssprog.

I forbindelse med formuleringen af generel systemmodellering her er den stigende udbredelse af mønstre og disses relation til generel systemteori kort berørt med henblik på at et generelt systemmodelleringssprog også skal kunne understøtte den form for formaliserede og genanvendelige systemmodeller. Generel systemteori kan dog eventuelt være af meget større betydning for mønstre end dette.

Mønstre har primært et pragmatisk udgangspunkt som specifikationer af løsninger der i praksis har vist sig som værende velegnede i forbindelse med tilbagevendende problemer. De forskellige identificerede systemtyper og systemegenskaber fra generel systemteori

samt mere specialiserede systemteoretiske fagområder som cybernetik kan eventuelt danne et alternativt udgangspunkt for en bredere, grundigere og mere systematisk identificering og beskrivelse af mønstre. Mønstre og generel systemteori kan formodentligt føre til en væsentlig bedre forståelse af komplekse systemer på et højere abstraktionsniveau end det har været tilfældet med traditionel systemudvikling.

Blød og kritisk systemteori har ligeledes været kort berørt i forbindelse med formuleringen af generel systemmodellering her med henblik på at et generelt systemmodelleringssprog skal kunne understøtte uformelle og ad hoc systemmodeller og systemudviklingsforløb. Med fokus på de reelle behov for systemer og systemudvikling samt selve systemudviklingsprocessen har disse potentielt meget stor betydning for systemudvikling. Blød og kritisk systemteori er dog også allerede i vid udstrækning inddraget i nogle systemudviklingsmetodologier.

Generel systemmodellering som formuleret her er således langtfra løsningen på alle problemer i forbindelse med systemudvikling af computerbaserede systemer, men forhåbentligt kan det være udgangspunkt for at mindske nogle centrale, unødvendige forhindringer for systemudviklingsforløb, således at systemudviklerne i højere grad kan fokusere på de essentielle problemer og behov i forhold til brugerne i stedet for systemmodelleringen.

Endvidere kan formuleringen af generel systemmodellering eventuelt motivere til i stigende grad at søge løsninger i forbindelse med systemudvikling af computerbaserede systemer indenfor generel systemteori og andre systemteoretiske fagområder. Det er formodentligt nødvendigt med mange, specialiserede metoder og værktøjer for at opbygge de nødvendige erfaringer, men efterhånden som videnen udbygges er der eventuelt behov og mulighed for at generalisere løsningerne.

Referencer

prev

Abowd, Gregory & Allen, Robert & Garlan, David (1993). Using Style to Understand Descriptions of Software Architecture, ACM SIGSOFT Software Engineering Notes, Vol. 18, No. 5, Dec. 1993 & SIGSOFT '93. Proceedings of the first ACM symposium on Foundations of software engineering

Agerholm, Sten & Larsen, Peter Gorm (1998). A Lightweight Approach to Formal Methods, Proceedings of the International Workshop on Current Trends in Applied Formal Methods, Springer-Verlag.

Andersen, Niels Erik & Kensing, Finn & Lassen, Monika & Lundin, Jette & Mathiassen, Lars & Munk-Madsen, Andreas & Sørgaard, Pål (1986). Professional Systemudvikling, Teknisk Forlag.

Appel, Andrew W. (1998). Modern compiler implementation in Java, Cambridge University Press.

Ashcroft, Ed & Wadge, Bill (1985). Lucid Primer, fra Lucid, the Dataflow Programming Language. Academic Press.

Avison, D. E. & Fitzgerald, G. (1995). Information Systems Development, McGraw-Hill International (UK).

Avison, D. E. & Wood-Harper, A. T. (1990). Multiview, Blackwell Scientific Publications.

Bakker, J. W. de & Roever, W.-P. de & Rozenberg, G. (1994). A Decade of Concurrency, Lecture Notes in Computer Science Vol. 803, Springer-Verlag.

Baldwin, Reid & Chung, Moon Jung (1994). Design Methodology Management Using Graph Grammars, DAC '94. Proceedings of the 31st annual conference on Design Automation.

Baresi, Luciano & Pezzé (1998). Toward Formalizing Structured Analysis, ACM Transactions on Software Engineering and Methodology, Vol. 7, No. 1, January 1998.

Bassett, Paul G. (1987). Frame-Based Software Engineering, IEEE Software, Vol. 4, No. 4, July 1987 & i DeMarco & Lister (1990).

Battista, Giuseppe Di & Eades, Peter & Tamassia, Roberto & Tollis, Ioannis G. (1999). Graph Drawing: Algorithms for the Visualization of Graphs, Prentice-Hall.

Bertalanffy, Ludwig von (1968). General System Theory, George Braziller.

Bézivin, Jean (1998). The sNet Formalism - Technical Report, http://www.sciences.univ-nantes.fr/info/lrsg/sNets/index.html.

Bézivin, Jean (1998). Who's Afraid of Ontologies?, OOPSLA'98 Workshop #25.

Boehm, Barry W. (1988). A Spiral Model of Software Development and Enhancement, IEEE Computer, Vol. 21, May 1988.

Boehm, Barry W. & Penedo, Maria H. & Stuckle, Don & Williams, Robert D. & Pyster, Arthur B. (1984). A Software Development Environment, IEEE Computer, Vol. 17, No. 6, June 1986 & i DeMarco & Lister (1990).

Boulding, Kenneth E. (1956). General Systems Theory - The Skeleton of Science, Management Science, 2, & i Klir (1991).

Brinksma, E. (1997). Tools and Algorithms for the Construction and Analysis of Systems (TACAS'97) Workshop, Lecture Notes in Computer Science, Springer-Verlag.

Brooks, jr., Fredrick P. (1995, 1975). The Mythical Man-Month, Addison Wesley Longman.

Brooks, jr., Fredrick P. (1986). No Silver Bullet, i: Kugler, H. J. (1986) Information Processing 86, Elsevier Science (North Holland) & IEEE Computer, Vol. 20, No. 4, April 1987.

Brooks, jr., Fredrick P. (1995). "No Silver Bullet" Refired, i Brooks (1995, 1975).

Brown, John Seely & Duguid, Paul (1996). Keeping It Simple, i Winograd (1996).

Brown, William J. & Malveau, Raphael C. & McCormick III, Hays W. "Skip" & Mowbray, Thomas J. (1998). Anti Patterns: Refactoring Software, Architectures, and Projects in Crisis, John Wiley & Sons.

Butler, Keith A. & Esposito, Chris & Hebron, Ron, Connecting the Design of Software to the Design of Work, Communications of the ACM, Vol. 42, No. 1, January 1999

CDIF Technical Committee (1994). CDIF Framework for Modeling and Extensibility, Electronic Industries Association.

Champeaux, Dennis de & Lea, Douglas & Faure, Penelope (1993). Object-Oriented System Development, Addison-Wesley.

Charette, Robert N. (1987). Software Engineering Environments: Concepts and Technology, McGraw-Hill.

Checkland, Peter & Scholes, Jim (1990). Soft Systems Methodology in Action, John Wiley & Sons.

Christensen, Søren & Kreiner, Kristian (1994). Projektledelse i løst koblede systemer, Jurist- og Økonomforbundets Forlag.

Ciancarini, Paolo (1996). Coordination Models and Languages as Software Integrators, ACM Computing Surveys, Vol. 28, No. 2, June 1996.

Clarke, Edmund M. & Wing, Jeanette, M. (1996). Formal Methods: State of the Art and Future Directions, ACM Computing Surveys, Vol. 28, No. 4, December 1996.

Consens, Mariano & Mendelzon, Alberto & Ryman, Arthur (1992). Visualizing and Querying Software Structures, ICSE '92, Proceedings of the 14th international conference on Software engineering.

Curtis, Bill & Kellner, Marc I. & Over, Jim (1992). Process Modeling, Communications of the ACM, Vol. 35, No. 9, September 1992.

Dahlbom, Bo & Mathiassen, Lars (1993). Computers in Context: The Philosophy and Practice of Systems Design, NCC Blackwell.

Date, C. J. & Darwen, Hugh (1997). A Guide to the SQL Standard, Addison-Wesley Longman.

Davis, Margaret J. & Williams, Roger B. (1997). Software Architecture Characterization, ACM SIGSOFT Software Engineering Notes, Vol. 22, No. 3, May 1997 & SSR '97, Proceedings of the 1997 Symposium on Software Reusability

DeMarco, Tom & Lister, Timothy (1990). Software State-of-the-Art, Dorset House Publishing.

Ellis, Keith & Gregory, Amanda & Mears-Young, Bridget R. & Ragsdell, Gillian (1995). Critical Issues in Systems Theory and Practice, Plenum Press.

Feijs, Loe & De Jong, Roel (1998). 3D Visualization of Software Architectures, Communications of the ACM, Vol. 41, No. 12, December 1998.

Feyerabend, Paul (1995, 1988, 1975). Against Method, Verso.

Finkel, Raphael A. (1996). Advanced Programming Language Design, Addison-Wesley Publishing Company.

Fjällström, Per-Olof (1998). Algorithms for Graph Partitioning: A Survey, Linköping Electronic Articles in Computer and Information Science, Vol. 3, nr 10, 1998, Linköping University Electronic Press.

Flood, Robert L. & Jackson, Michael C. (1991). Critical Systems Thinking, John Wiley & Sons.

Fowler, Martin (1997). Analysis Patterns: Reusable Object Models, Addison Wesley Longman.

France, Robert B. (1992). Semantically Extended Data Flow Diagrams: A Formal Specification Tool, IEEE Transactions on Software Engineering, Vol. 18, No. 4, April 1992.

Fraser, Martin D. & Kumar, Kuldeep & Vaishnavi, Vijay K. (1991). Informal and Formal Requirements Specification Languages: Bridging the Gap, IEEE Transactions on Software Engineering, Vol. 17, No. 5, May 1991.

Gal, Shaf (1996). Footholds for Design, i Winograd (1996).

Gamma, Erich & Heml, Richard & Johnson, Ralph & Vlissides, John (1995). Design Patterns: Elements of Reusable Object-Oriented Software, Addison Wesley Longman.

Gelernter, David & Carriero, Nicholas (1992). Coordination Languages and their Significance, Communications of the ACM, Vol. 35, No. 2, February 1992.

Goldman, Kenneth J. & Anderson, Michael D. & Swaminathan, Bala (1994). The Programmers' Playground: I/O Abstraction for Heterogeneous Distributed Systems, IEEE Transactions on Software Engineering, Vol. 21 No. 9, September 1995.

Harel, David (1998). On Visual Formalisms, Communications of the ACM, Vol. 31, No. 5, May 1998.

Heimdahl, Mats P.E. & Leveson, Nancy G. (1995). Completeness and Consistency Analysis of State-Based Requirements, ICSE '95. Proceedings of the 17th International Conference on Software engineering.

Heitmeyer, Constance L. & Jeffords, Ralph D. & Labaw, Bruce G. (1996). Automated Consistency Checking of Requirements Specifications, ACM Transactions on Software Engineering and Methodology, Vol. 5, No. 3, July 1996

Howes, Timothy A. & Smith, Mark C. & Good, Gordon S. (1999). Understanding and Deploying LDAP Directory Services, Macmillan Technical Publishing.

Jackson, Michael (1983). System Development, Prentice-Hall International.

Jackson, Michael (1995). Software Requirements & Specifications, ACM Press.

Jarke, Matthias (1998). Requirements Tracing, Communications of the ACM, Vol. 41, No. 12, December 1998

Jensen, Kurt (1997). A Brief Introduction to Coloured Petri Nets, i Brinksma (1997).

Jensen, Kurt (1996). An Introduction to the Practical Use of Coloured Petri Nets, i Reisig & Rozenberg (1996).

Jensen, Kurt (1994). An Introduction to the Theoretical Aspects of Coloured Petri Nets, i Bakker & Roever & Rozenberg (1994).

Kapur, Deepak (1994). An Automated Tool for Analyzing Completeness of Equational Specifications, ISSTA '94. Proceedings of the 1994 International Symposium on Software Testing and Analysis.

Kim, Won & Lochovsky, Frederick H. (1989). Object-Oriented Concepts, Databases, and Applications, ACM Press.

Kleyn, M. F. & Browne, J. C. (1993). A High Level Language for Specifying Graph Based Languages and their Programming Environments, ICSE '93, Proceedings of the 15th international conference on Software Engineering.

Klir, George J. (1991). Facets of Systems Science, Plenum Press.

Knuth, D. E. (1973). The Art of Computer Programming, Vol. 1: Fundamental Algorithms, Addison-Wesley.

Kuhn, Sarah (1996). Design for People at Work, i Winograd (1996).

Kummer, Stephan D. & Schlange, Lutz E. (1997). Strengthening the Bridge between Qualitative and Quantitative Modeling, 15th International System Dynamics Conference.

Lanergan, Robert G. & Grasso, Charles A. (1984). Software Engineering with Reusable Design and Code, IEEE Transactions on Software Engineering, Vol SE-10, No. 5, September 1984 & i DeMarco & Lister (1990).

Larsen, Peter Gorm & Katwijk, Jan van & Plat, Nico & Pronk, Kees & Toetenel, Hans (1991). SVDM: An Integrated Combination of SA and VDM, Methods Integration Conference, Leeds, September 1991.

Larsen, Peter Gorm & Plat, Nico & Toetenel, Hans (1993). A Formal Semantics of Data Flow Diagrams, Formal Aspects of Computing, 3, 1993.

Ledbetter, Lamar & Cox, Brad (1985). Software-ICs: A Plan for Building Reusable Software Components, Byte, Vol. 10, No. 6, June 1985 & i DeMarco & Lister (1990).

Lemesle, Richard (1998). Meta-modeling and Modularity, OOPSLA'98 Workshop #25.

Lung, Chung-Horng (1998). Software Architecture Recovery and Restructuring through Clustering Techniques, ISAW '98, Proceedings of the third international workshop on Software architecture.

McCartney, T. Paul (1996). End-User Construction and Configuration of Distributed Multimedia Applications, D. Sc. Thesis, Washington University.

McHugh, James A. (1990). Algorithmic Graph Theory, Prentice Hall.

Medvidovic, Nenad & Taylor, Richard N. (1998). Separating Fact from Fiction in Software Architecture, ISAW '98, Proceedings of the third international workshop on Software architecture.

Métayer, Daniel Le (1996). Software Architecture Styles as Graph Grammars, ACM SIGSOFT Software Engineering Notes, Vol. 21, No. 6, Nov. 1996 & SIGSOFT '96, Proceedings of the fourth ACM SIGSOFT symposium on Foundations of software engineering.

Mintzberg, Henry (1983). Structure in Fives: Designing Effective Organizations, Prentice-Hall.

Morrison, J. Paul (1994). Flow-Based Programming, Van Nostrand Reinhold.

Motorola Inc. (1986). M68000 8-/16-/32-Bit Microprocessors Reference Manual, Motorola Inc.

Mylopoulos, John & Chung, Lawrence & Yu, Eric (1999). From Object-Oriented to Goal-Oriented Requirements Analysis, Communications of the ACM, Vol. 42, No. 1, January 1999

Norman, Donald (1996). Design as Practiced, i Winograd (1996).

Oberweis, Andreas & Sander, Peter (1996). Information System Behavior Specification by High-Level Petri Nets, ACM Transactions on Information Systems, Vol. 14, No. 4, October 1996.

OMG/Component Scripting (1998). CORBA Component Scripting, Revised Joint Submission, July 6th, 1998, Object Management Group.

OMG/CORBA (1998). The Common Object Request Broker: Architecture and Specification, Object Management Group.

OMG/CORBA (1995). CORBAfacilities: Common Facilities Architecture, Object Management Group.

OMG/CORBA (1997). CORBAservices: Common Object Services Specification, Object Management Group.

OMG/MOF (1997). Meta Object Facility (MOF) Specification, Joint Revised Submission, September 1, 1997, Object Management Group.

OMG/UML (1997). Unified Modelling Language, version 1.1, 1. September, 1997, Object Management Group.

Parnas, David Lorge & Clements, Paul C. (1986). A Rational Design Process: How and Why to Fake It, IEEE Transactions on Software Engineering, Vol. SE-12, No. 2, February 1986 & i DeMarco & Lister (1990).

Plat, Nico & Larsen, Peter Gorm & Toetenel, Hans (1993). Formal Transformations: Using SA and VDM as Different Views in Software Development, Technical paper, The Institute of Applied Science (IFAD) & Delft University of Technology.

Reisig, Wolfgang & Rozenberg, Grzegorz (1998): Lectures on Petri Nets II: Applications, Advances in Petri Nets, Lecture Notes in Computer Science, Springer-Verlag.

Rheinfrank, John & Evenson, Shelley (1996). Design Languages, i Winograd (1996).

Roman, Gruia-Catalin & Cox, Kenneth C. (1992). Program Visualization: the Art of Mapping Programs to Pictures, ICSE '92. Proceedings of the 14th International Conference on Software Engineering.

Royce (1970). Managing the Development of Large Software Systems, Proceedings of the IEEE WESCON, IEEE Press & Proceedings of the Ninth International Conference on Software Engineering, IEEE Press.

Rozenberg, G (1997). Handbook of Graph Grammars and Computing by Graph Transformation: Foundations, Vol. 1, World Scientific Publishing Company.

Saffo, Paul (1996). The Consumer Spectrum, i Winograd (1996).

Schön, Donald & Bennett, John (1996). Reflective Conversation with Materials, i Winograd (1996).

Sommerville, Ian (1992). Software Engineering, Addison-Wesley Publishing Company.

Stowell, Frank (1995). Does 'Information Systems' Need Systems?, i Ellis, Gregory, Mears-Young & Ragsdell (1995).

Tanenbaum, Andrew S. (1992). Modern Operating Systems, Prentice-Hall.

Tanenbaum, Andrew S. (1987). Operating Systems, Prentice-Hall.

Tanenbaum, Andrew S. (1990, 1984, 1976). Structured Computer Organization, Prentice-Hall.

Wasserman, Stanley & Faust, Katherine (1994). Social Network Analysis: Methods and Applications, Cambridge University Press.

Weinberg, Gerald M. (1988). General Principles of Systems Design, Dorset House Publishing.

Weinberg, Gerald M. (1975). An Introduction to General Systems Thinking, John Wiley & Sons.

Weinberg, Gerald M. (1988). Rethinking Systems Analysis & Design, Dorset House Publishing.

Wilson, Brian (1990, 1984). Systems: Concepts, Methodologies and Applications, John Wiley & Sons.

Winograd, Terry (1996). Bringing Design to Software, ACM Press.

Wirth, N. (1976). Algorithms & Data Structures = Programs, Prentice-Hall.

Yourdon, Edward (1989). Modern Structured Analysis, Prentice-Hall.

Zave, Pamela & Jackson, Michael (1997). Four Dark Corners of Requirements Engineering, ACM Transactions on Software Engineering and Methodology, Vol. 6, No. 1, January 1997.

Zeigler, Bernard P. (1976). Theory of Modelling and Simulation, John Wiley & Sons.

Zeigler, Bernard P. & Vahie, Sankait (1993). DEVS Formalism and Methodology, Proceedings of the 1993 Winter Simulation Conference.

Züllighoven, Heinz (1998). The Tools & Materials Approach for Large Scale OO Software Development, JAOO '98 Conference, Copenhagen.

Centrale problemer ved systemudvikling

next

Systemudviklingsfilosofi

Systemopfattelse

Systemudviklingsformål

Systemudviklingsforløb

Systemudviklingsparadig-mer

Systemudviklingsmetodol-ogier

Systemmodellering

Systemudviklingsværktøje-r

De følgende afsnit præsenterer en bred analyse af alle dele af og aspekter ved et komplet systemudviklingsforløb. Analysen omfatter grundlæggende antagelser såvel som konkrete systemudviklingsmetodologier, -metoder og -værktøjer.

Udgangspunktet for analysen er "state-of-practice" med almindeligt kendte og udbredte systemudviklingsmetodologier, -metoder og -værktøjer, og de her identificerede problemer kendetegner således primært den "typiske" systemudvikling der foregår. Specifikke systemudviklingsteorier og systemudviklingspraksiser adresserer allerede (nogle af) de forskellige identificerede problemer, men analysen her er begrænset til en (grov) generalisering af typisk systemudvikling med henblik på at identificere tværgående generelle problemer.

Analysen omfatter de følgende dele af og aspekter ved systemudviklingsforløb:

Den bagvedliggende filosofi for systemudvikling er typisk ikke eksplicit formuleret i forbindelse med forskellige systemudviklingsmetodologier, -metoder og -værktøjer, men ved valg mellem sådanne er det vigtigt om systemudviklingsfilosofien er i overensstemmelse med ens egne grundlæggende antagelser.

Den grundlæggende opfattelse af systemer i forbindelse med forskellige systemudviklingsmetodologier, -metoder og -værktøjer er typisk ligeledes heller ikke eksplicit formuleret men vigtig for egnetheden til den konkrete systemudviklingsopgave.

Systemudviklingsopgaver varierer meget med hensyn til det grundlæggende formål med systemudviklingsforløbet fra engangs konstruktion af et teknisk system; over en evolutionær proces med udvikling af et system og omgivelserne; til systemudvikling som sekundær proces for udvikling af omgivelserne.

Systemudviklingsfilosofi, systemopfattelse og systemudviklingsformål afspejles i forskellige procesmodeller for systemudviklingsforløb.

Ved konstruktion af tekniske systemer anvendes der en række forskellige systemudviklingsparadigmer med udgangspunkt i og fokus på forskellige aspekter ved systemet og systemudviklingsprocessen.

Som udgangspunkt for systemudvikling anvendes der ofte en af de mange forskellige foreslåede systemudviklingsmetodologier med retningslinier for "god" systemudvikling.

Kernen i systemudvikling er systemmodellering fra forståelse af problemet og løsningen til det egentlige, implementerede system som i sig selv (delvist) er en eksekverbar model i form af et computerbaseret system.

Udvikling af ikke trivielle computerbaserede systemer kræver meget omfattende og kompleks systemodellering, der er umulig uden anvendelse af systemudviklingsværktøjer knyttet til de forskellige systemudviklingsmetodologier og metoder.

Systemudviklingsfilosofi

next up

Feyerabend [1975, 1988, 1993] Against Method

Parnas & Clements [1986] A Rational Design Process: How and Why to Fake It

Den bagvedliggende filosofi for hovedparten af systemudvikling er ikke eksplicit formuleret, men det ses tydeligt at den hovedsageligt tager udgangspunkt i et klassisk videnskabeligt rationalitets ideal om en rationelt fremadskridende proces igennem problemanalyse, teoriformulering, implementering og test afsluttet med et sandhedsudsagn og eventuel accept og ibrugtagning.

På tilsvarende måde bygger systemudvikling på en ide om et grundlæggende, rationelt fremadskridende systemudviklingsforløb igennem analyse, design, implementering, test og drift af systemer eller en tilsvarende faseopdeling af systemudviklingsforløbet. Dette gælder specielt for den klassiske "vandfaldsmodel" for systemudvikling, men grundlæggende set gælder det også for nyere modeller for systemudviklingsforløb. I disse er den klassiske "vandfaldsmodel" dog tilpasset til at omfatte flere iterationer igennem en cirkel eller spiral, som følge af at det i praksis har vist sig meget vanskeligt at nå frem til en "rigtig" løsning på et enkelt gennemløb. Indenfor de enkelte gennemløb er der dog grundlæggende stadig tale om et rationelt fremadskridende forløb igennem faserne.

Forestillingen om det rationelt fremadskridende videnskabelige arbejde afvises af Feyerabend, der i stedet for mener at kunne vise at selv nogle af videnskabens store forbilleder afviger kraftigt fra idealet. Det videnskabelige arbejde og de videnskabelige resultater involverer en tæt vekslen mellem problemanalyse, teoriformulering og implementering/test med gensidig påvirkning mellem de forskellige faser, og der indgår en væsentlig grad af tilfældigheder, intuition, retorik og politik. Der kan således ikke hævdes at være én sand, rationelt fremadskridende videnskabelig fremgangsmåde.

Den dominerende opfattelse af systemudvikling som en rationelt fremadskridende proces passer også dårligt med virkelighedens praksis, hvor systemudviklere i vid udstrækning må omgå/reparere på forudsætningerne i de anvendte metoder og værktøjer. Der er gode grunde til at dokumentere/præsentere resultaterne af systemudviklingsforløb som rationelt fremadskridende, men det er en illusion at tro at systemudviklingsforløbet selv kan afspejle dette ideal. Det er hensigtsmæssigt/nødvendigt at systemudviklingsprocessen og systemudviklingsproduktet ikke forudsætter hinanden.

Ønsket om at finde én generelt anvendelig, rationelt fremadskridende systemudvikling fremgangsmåde er således formodentligt uopnåelig. I stedet for bør generelle systemudviklingsmetoder og værktøjer være forberedt på mange og meget forskellige systemudviklingsforløb. Dermed dog ikke sagt at mere snævre/ensrettede systemudviklingsmetoder og værktøjer ikke er mulige og anvendelige til bestemte typer eller dele af systemudvikling. Tværtimod, det vil i høj grad være tilfældet, men det er uholdbart at forsøge at tvinge alt ind i faste forløb. Generel systemudvikling bør kunne rumme alle tænkelige systemudviklingsforløb - "tilfældige" såvel som "standardiserede" og kombinationer.

Systemopfattelse

prev next up

Dahlbom & Mathiassen [1995] Computers in Context

Systemudvikling som den praktiseres idag er tydeligt vis udsprunget med udgangspunkt i computer teknologien. Fra oprindeligt at fokusere på computer hardwaren og maskinkode programmering er abstraktionsniveauet igennem årene blevet hævet fra computer hardware og maskinkode over assemblersprog og højniveau programmeringssprog til design modelleringssprog, og der er inddraget stadig mere i systemudviklingen fra snæver programmering og programafvikling til systemdesign, systemanalyse og problemanalyse samt organisatorisk implementering, drift og vedligeholdelse.

Den tidlige systemudvikling og måske stadig hovedparten af den nuværende systemudvikling sker med udgangspunkt i "hard systems" tænkning, hvor fokus i vid udstrækning i specielt systemudviklingsprocessen men også produktet er den tekniske løsning.

Senere er systemudviklingen i vid udstrækning inddraget i en personlig/organisatorisk kontekst med udgangspunkt i specielt "soft systems" tænkning samt "dialectic systems" tænkning. Dette har medført mange nye og ændrede ting i forbindelse med systemudvikling, men det er primært i forhold til formålet med systemudvikling samt problemanalyse og systemanalyse. "Soft systems" og "dialectic systems" tænkning har bidraget med nye lag på systemudviklingen, men når det kommer til udviklingen af den tekniske løsning, er det i overvejende grad stadig de "gamle" systemudviklingsmetoder og -værktøjer fra "hard systems" tænkning, der anvendes. Dette må nødvendigvis give en del begrænsninger i effekten af "soft systems" og "dialectic systems" tænkning.

Der er således ikke foretaget en gennemgående revidering af det samlede systemudviklingsforløb men primært repareret og bygget videre på det gamle fundament. Med så mange og omfattende ændringer og udbygninger af systemudvikling igennem tiden kunne der være et stort udbytte ved en gennemgående revidering.

Systemudviklingsforløbet skal fortsat ende ud i en teknisk løsning men selv de meget maskin-nære ting kan måske laves på en måde, der stadig er effektiv, men som passer bedre til systemudvikling i "soft systems" og "dialectic systems" kontekst, således at det samlede systemudviklingsforløb eventuelt kan foregå på en mere hensigtsmæssig måde.

Systemudviklingsformål

prev next up

Dahlbom & Mathiassen [1995] Computers in Context

Avison & Fitzgerald [1995] Information Systems Development

Traditionel systemudvikling er foregået som konstruktion af et system udfra specifikationer med efterfølgende drift og vedligeholdelse af systemet i begrænset omfang. Eventuelle større ændringer og tilføjelser er lavet som endnu et systemudviklingsforløb med konstruktion af et nyt system baseret på det eksisterende. Systemer konstrueres til at være færdige og potentielt at skulle anvendes i en længere periode.

For at håndtere ukendte krav og løbende forandringer i mange projekter sker systemudviklingen i stedet for som en mere evolutionær proces med gennemløb af et antal iterationer med gradvis ændring/udbygning og test af systemet. Målet er dog stadig at udvikle et færdigt system som løsning på et givent problem.

Systemet, der udvikles, er ikke nødvendigvis målet med systemudviklingen, men det kan primært være en anledning og et middel til at opnå forandringer i de omkringliggende systemer, og selve systemet er kun sekundært.

Traditionel systemudvikling er dårligt egnet til håndtering af ukendte og foranderlige krav til systemet, men selv med kendte og uforanderlige krav kan det være problematisk at specificere kravene og sikre en korrekt implementering med de traditionelle systemudviklingsmetoder og -værktøjer, samt at videreudvikle systemet.

Der er lavet mange tiltag for at tilpasse traditionel systemudvikling til en evolutionær process for eksempel i form af Boehms spiralmodel for systemudviklingsprocessen samt prototyping og Rapid Application Development (RAD) metoder og værktøjer. Der er dog stadig en del problemer i forbindelse med prototyping og RAD idet de udviklede systemer ofte er ineffektive, ukomplette og dårligt dokumenterede, hvorfor de ikke er egnede til store systemer medmindre, at de efterfølgende genudvikles med de traditionelle metoder og værktøjer. Desuden er prototyping og RAD metoder og værktøjer ofte primært beregnet til bestemte typer af systemer som for eksempel administrative database-baserede løsninger. Endeligt er den typiske, eksisterende evolutionære systemudvikling også primært beregnet til at nå et færdigt resultat og ikke så meget til en eventuelt uendelig udviklingsproces med konstant, løbende videreudvikling af systemet, da systemet eventuelt efterfølgende genimplementeres med mere driftseffektive værktøjer.

Med udviklingsprocessen selv og omkringliggende forandringer som det primære produkt er eksisterende systemudvikling forholdsvis rigid, omfattende og kompliceret for at få de tekniske systemer inddraget. Det begrænser mulighederne for bred og dyb problemudforskning, og der mangler fokus på understøttelse af flertydighed, alternative synsvinkler og konflikterende interesser. Håndtering af dette må ske med isolerede metoder og værktøjer ved siden af de almindelige teknisk orienterede metoder og værktøjer.

Generel systemmodellering bør således understøtte den nødvendige detaljering og præcision til specifikation og implementering af pålidelige og effektive computer systemer. Samtidigt bør generel systemmodellering være fleksibel og effektiv til eksperimenteren med håndtering af uklarheder, alternativer og konflikter; hurtig implementering af ad-hoc systemer og løbende implementering af computer systemer under konstant forandring. Endeligt bør generel systemmodellering have den nødvendige fleksibilitet for overgang fra eksperimenterende og ad-hoc computer systemer til pålidelige og effektive computer systemer (og omvendt). Dermed bør der ikke være grundlæggende forskel på generel systemmodellering i forbindelse med forskellige systemudviklingsformål.

Systemudviklingsforløb

prev next up

Royce [1970] Managing the development of large software systems

Yourdon [1989] Modern Structured Analysis

Boehm [1988] A spiral model of software development and enhancement

Checkland [1990] Soft Systems Methodology in Action

Avison & Wood-Harper [1990] Multiview

Andersen, Kensing, Lassen, Lundin, Mathiassen, Munk-Madsen & Sørgaard [1986] Professionel Systemudvikling

Schön & Bennett [1996] Reflective Conversation with Materials

Gal [1996] Footholds for Design

Traditionel systemudvikling sker gennem et antal sekventielle faser efter en såkaldt "vandfaldsmodel", hvor resultatet af hver enkelt fase er et afsluttet produkt, der danner udgangspunkt for den efterfølgende fase. Faserne kan typisk være kravspecifikation, analyse, design, programmering, test og drift. Der er således tale om et strengt rationelt, ensrettet systemudviklingsforløb.

Vandfaldsmodellen er i vid udstrækning blevet erstattet af for eksempel struktureret systemudvikling, som populariseret af Yourdon, hvor de sekventielle faser erstattes af tilsvarende aktiviteter. I modsætning til det traditionelle faseforløb, kan aktiviteterne ske (delvist) parallelt og iterativt, således at man ikke længere er afhængig af at det lykkes at lave et korrekt produkt i ét forsøg som resultat af de enkelte faser. Grundlæggende er der dog stadig tale om et rationelt, ensrettet systemudviklingsforløb, hvor resultaterne af de enkelte aktiviteter danner udgangspunkt for de efterfølgende aktiviteter.

Struktureret systemudvikling resulterer som traditionel systemudvikling i ét produkt som resultat af hver enkelt aktivitet, hvilket giver begrænsede muligheder for styring af systemudviklingsforløbet. Boehm opstiller som alternativ et mere evolutionært systemudviklingsforløb i form af en spiralmodel med produkter som resultat af hver iteration, hvor hver iteration sker indenfor de overordnede fire faser: fastlæggelse af mål, alternativer og begrænsninger; evaluering af alternativer samt identificering og håndtering af risici; udvikling og efterprøvning af næste produkt/udgave; planlægning af næste fase. Spiralmodellen er således en overordnet styringsmodel, der typisk rummer en passende sammensætning af traditionel og/eller struktureret systemudvikling suppleret med prototyping til risikoafklaring.

Såvel traditionel og struktureret systemudvikling som spiralmodellen er fokuserede på at levere et færdigt teknisk system, som resultat af systemudviklingsforløbet. Checkland's Soft Systems Methodology opstiller derimod et systemudviklingsforløb, som en principielt uendelig læringsproces med idenfitikation og formulering af problemsituation; formulering af alternative systemmodeller; valg af ønskede ændringer; samt handling for at forbedre problemsituationen. Dette er således også en overordnet læringsmodel, der som for eksempel i Multiview må suppleres med et andet systemudviklingsforløb med fokus på at implementere det tekniske system, hvis sådanne indgår i problemløsningen.

Fælles for systemudviklingsforløbene - direkte i traditionel og struktureret systemudvikling og eventuelt igennem det underliggende teknisk orienterede systemudviklingsforløb i spiralmodellen og Soft Systems Methodology - findes et grundlæggende, rationelt, ensrettet forløb fra analyse over design til implementering, hvilket der kan/bør stilles spørgsmålstegn ved. Andre formulerer tætte gensidige relationer mellem analyse, design og implementering; igennem design og implementering identificeres forhold til analyse og igennem implementering identificeres designmuligheder og -problemer. Struktureret systemudvikling kan hævdes at reflektere dette igennem iterationen, men dette er dog ikke tilstrækkeligt, idet resultaterne af de enkelte aktiviteter klart betinger de efterfølgende aktiviteter selvom det kan ske i flere forsøg. Dette afspejles ikke mindst i de tilknyttede metoder og værktøjer. Spiralmodellen giver derimod mulighed for at rumme ikke-ensrettede systemudviklingsforløb.

Generel systemudvikling bør kunne rumme meget forskellige systemudviklingsforløb - såvel teknisk orienterede som (risiko)styrings- og læringsorienterede systemudviklingsforløb (også andre end de her beskrevne) - uden givne antagelser om antallet og rækkefølgen af aktiviteter/faser med fri mulighed for at sammensætte et vilkårligt systemudviklingsforløb med analyse, design og implementering aktiviteter.

Systemudviklingsparadigmer

prev next up

Avison & Fitzgerald [1995] Information Systems Development

Sommerville [1992] Software Engineering

Schön & Bennett [1996] Reflective Conversation with Materials

De tre primære, overordnede indgangsvinkler til analyse, design og implementering af software er funktionel, data- og objektorienteret systemudvikling med fokus på henholdsvis processerne og trinvis nedbrydning/forfining af disse; information-/datastrukturer i form af elementer og relationer mellem disse; og gruppering af "naturligt" sammenhørende funktioner og data i selvstændige objekter og relationer mellem disse.

De forskellige synsvinkler/fremgangsmåder er velegnede til forskellige opgaver afhængigt af det enkelte projekt, den enkelte udvikler og de enkelte aktiviteter indenfor et projekt. Typisk er man dog bundet til ét enkelt paradigme afhængigt af valget af systemudviklingsmetodologi/-værktøj indenfor projektet/virksomheden.

Er der reelt tale om uforenelige paradigmer med fundamentale forskelle eller er det blot dårlig implementering når det ikke er muligt eller vanskeligt at kombinere de forskellige indgangsvinkler? I sidste ende drejer det sig trods alt om de samme data, funktioner og relationer mellem disse.

Med generel systemmodellering bør det være muligt frit at kunne skifte mellem de forskellige synsvinkler, således at den bedst egnede indgangsvinkel kan vælges til den enkelte delopgave for den enkelte udvikler. Ikke blot således at der kan vælges forskellige synsvinkler på de enkelte delopgaver, men således at hver enkelt delopgave frit kan præsenteres udfra flere/alle synsvinkler for at fremhæve forskellige aspekter.

En anden dimension på analyse, design og implementering af software er henholdsvis top-down og bottom-up baserede systemudviklingsforløb. Ved top-down systemudvikling nedbrydes problemstillingen analytisk og trinvist indtil et niveau bestående af simple komponenter og programinstruktioner. Ved bottom-up systemudvikling løses problemstillingen konstruktivistisk med kombination af programinstruktioner og simple komponenter til trinvist større komponenter indtil et komplet system nåes.

Top-down og bottom-up baseret systemudvikling betegnes ofte som uforenelige paradigmer og forskellige systemudviklingsmetodologier, -metoder og -værktøjer tager udgangspunkt i et af dem. I forbindelse med generel systemmodellering bør top-down og bottom-up baseret systemudvikling dog frit kunne kombineres, da de i stedet for uforenelige paradigmer kan ses som naturlige, supplerende indgangsvinkler der er velegnede til forskellige aktiviteter i et systemudviklingsforløb og forskellige dele af et system. Top-down baseret systemudvikling er velegnet til analytisk refleksion for problemstillinger og bottom-up er velegnet til eksperimentiel "dialog" med forskellige løsningsmuligheder.

Systemudviklingsmetodologier

prev next up

Avison & Fitzgerald [1995] Information Systems Development

Der findes et stort antal mere eller mindre forskellige systemudviklingsmetodologier indenfor de forskellige systemudviklingsparadigmer. Mange af disse "markedsføres" af en række forskellige organisationer og virksomheder med kommercielle interesser, som konsulenter eller software producenter. Nogle af de mest kendte er Yourdon's Yourdon Systems Method (struktureret systemudvikling); Martin's Information Engineering; Coad & Yourdon's ojektorienteret systemudvikling; Rumbaugh's OMT; Jacobson's OOSE; og Booch's objektorienterede systemudvikling; samt Martin's Rapid Application Development. Blandt mere forskningsorienterede systemudviklingsmetodologier er Soft Systems Methodology og Multiview blandt de mest kendte.

Der er således langtfra nogen anerkendt universel løsning, men mange forskellige metodologier, der hver især fungerer bedst indenfor forskellige rammer og begrænsninger, og det kan derfor være nødvendigt/ønskeligt at skifte systemudviklingsmetodologi efter behov i forbindelse med det enkelte systemudviklingsprojekt. Det kan også være hensigtsmæssigt at anvende forskellige metodologier indenfor et enkelt systems levetid. For eksempel indledende anvendelse af Information Engineering for implementering af en organisations centrale informationer; efterfølgende Rapid Application Development for implementering af supplerende funktioner; og afsluttende struktureret systemudvikling for udfasning/konvertering til et nyt system.

Systemudviklingsmetodologierne angiver de overordnede rammer for systemudviklingsforløbet i form af en række aktiviteter/faser med tilhørende metoder og værktøjer, men typisk kan der være behov for at tilpasninger til det enkelte systemudviklingsprojekt. På nogle områder kan metodologierne være for simple og firkantede, således at de må suppleres med andet for at være tilstrækkeligt dækkende. Omvendt kan de også være for store og komplicerede, således at unødvendige dele kan ignoreres.

Selvom der er en del fællestræk mellem mange systemudviklingsmetodologier, så er der dog også mange væsentlige forskelle. Systemudviklingsmetodologierne er typisk forholdsvis omfattende og komplicerede, og det er svært for systemudviklere at beherske forskellige metodologier (og ikke mindst dertil hørende systemudviklingsværktøjer). Systemudviklingsmetodologiernes produkter i form af diagrammer og specifikationer med mere er typisk også inkompatible, hvilket betyder at det er vanskeligt at kombinere og skifte mellem forskellige metodologier.

Uden udsigt til nogen universalløsning bør generel systemudvikling kunne rumme/understøtte de mange forskellige systemudviklingsmetodologier og specielle projekttilpassede varianter af disse. Det er desuden ønskeligt om generel systemudvikling kan medvirke til at integrere eller kombinere de forskellige metodologiers metoder og værktøjer for større valgmulighed og anvendelighed.

Systemmodellering

prev next up down

Domæne analyse

Problem analyse

Kravspecifikation

Systemanalyse

Systemdesign

Programmering

Maskinarkitektur

Avison & Fitzgerald [1995] Information Systems Development

Helt centralt i systemudvikling er de forskellige systemmodeller, der laves igennem systemudviklingsforløbet - fra modellering af problemområdet over kravspecifikation, analyse, design og programmering til selve det udviklede system i form af eksekverbare programmer og data samt eventuelle tilhørende manuelle procedurer.

Der anvendes meget forskellige modeltyper, der kan kategoriseres som verbale, matematiske, grafiske eller simulerede modeller, i de forskellige systemudviklingsmetodologier og indenfor de enkelte systemudviklingsmetodologier, og der anvendes modeller på meget forskellige abstraktionsniveauer i form af henholdsvis konceptuelle, logiske og fysiske modeller.

Resultatet af de forskellige faser/aktiviteter i systemudviklingsforløbet består i vid udstrækning af systemmodeller, som skal danne grundlaget for den videre systemudvikling/systemmodellering i de efterfølgende faser/aktiviteter. Typisk er der ikke nogen direkte sammenhæng mellem de forskellige systemmodeller i de forskellige faser/aktiviteter men kun indirekte sammenhæng i form af de metoder systemudviklingsmetodologierne angiver for systemudviklerne til at oversætte/fortolke og videreudvikle fra en model til en anden.

Anvendelsen af meget forskellige modeltyper på meget forskellige abstraktionsniveauer begrænser og komplicerer systemudviklingen:

Det er ikke umiddelbart muligt at skifte mellem forskellige modeltyper indenfor en enkelt fase/aktivitet til forskellige formål/situationer, som for eksempel anvendelse af en grafisk model til præsentation og overblik, en verbal model for generel forståelse, en matematisk model for detaljeret validering/verifikation og en simuleret model til afprøvning. Hvis der ønskes anvendelse af flere forskellige modeltyper skal de ofte laves som separate modeller med manuel sikring af konsistens mellem modellerne.

Den manglende/begrænsede direkte sammenhæng mellem modellerne i de forskellige faser/aktiviteter i systemudviklingsforløbet kan eventuelt ses som årsag til eller resultat af ensrettede systemudviklingsforløb. Metoderne angiver som regel kun oversættelse/fortolkning og videreudvikling i den ene "retning" i systemudviklingsforløbet ved at én fases/aktivitets systemmodeller danner rammerne for efterfølgende fasers/aktiviteters systemmodeller. Det er sjældent muligt direkte at arbejde i den modsatte retning af systemudviklingsforløbet, hvis der viser sig behov for at bryde rammerne. I sådanne tilfælde må man i stedet for iterere de relevante faser/aktiviteter i systemudviklingsforløbet, hvis der skal sikres konsistens i systemodellerne på tværs af faserne/aktiviteterne.

De mange forskellige systemudviklingsmetodologier anvender mange forskellige systemudviklingsmodeller selvom mange systemudviklingsmetodologier også anvender de samme systemudviklingsmodeller eller varianter af de samme grundlæggende systemudviklingsmodeller. Anvendelsen af forskellige systemudviklingsmodeller begrænser/umuliggør mulighederne for umiddelbart at skifte mellem forskellige systemudviklingsmetodologier.

Generel systemudvikling bør være baseret på tilsvarende generel systemmodellering baseret på fælles generelle koncepter og metoder på tværs af de forskellige systemudviklingsmetodologier, faser/aktiviter i systemudviklingsforløbet og formål/situationer, således at der igennem systemudviklingsforløbet opbygges én komplet (i det omfang det er nødvendigt/ønskeligt) og konsistent systemmodel.

Domæne analyse

next up

Yourdon [1989] Modern Structured Analysis

Mintzberg [1983] Structure in Fives

I forbindelse med systemudvikling af systemer med tætte og komplekse relationer til systemets tekniske og organisatoriske omgivelser kan det være nødvendigt/ønskeligt at foretage en bred og grundig domæne analyse for at erhverve den nødvendige viden til at forstå og fastlægge systemets relationer til omgivelserne. Der kan således være behov for at modellere meget forskellige typer af og aspekter ved systemets domæne(r).

Systemudviklingsmetodologier omfatter som regel ikke en decideret domæne analyse udover identificering af elementer i omgivelserme der har direkte relationer med systemet, samt specifikation af eksisterende processer der skal automatiseres eller forandres i forbindelse med systemudviklingen.

I det omfang systemudviklingsmetodologier omfatter domæne analyse sker modelleringen på stort set samme måde som systemmodelleringen. For eksempel omfatter (tidligere versioner) af struktureret systemudvikling modellering af det aktuelle fysiske og logiske system ved hjælp af dataflow diagrammer. Dette muliggør kun forholdsvis simpel domæne modellering, da det kun omfatter aspekter/elementer, der er relevante i forbindelse med computerbaserede systemer.

Til en teknisk domæne analyse af omkringliggende teknologier og produktionsprocesser kan en sådan simpel domæne modellering eventuelt være tilstrækkelig, men en bred og grundig domæne analyse bør dog kunne rumme mange andre aspekter/elementer, inklusiv ting der ikke er direkte relaterede til systemudvikling. En organisatorisk domæne analyse kunne omfatte en organisationsteoretisk organisationsstruktur analyse af formel autoritet, regulerede aktiviteter, uformel kommunikation, arbejdskonstellationer og ad hoc beslutningsprocesser.

Med generel systemudvikling bør det være muligt at inddrage domæne analyse direkte i systemudviklingen som en integreret del af systemmodelleringen således at der blandt andet kan sikres inddragelse af og sporbarhed af domæne forudsætninger for og konsekvenser af systemudviklingen.

Problem analyse

prev next up

Yourdon [1989] Modern Structured Analysis

Champeaux, Lea & Faure [1993] Object-Oriented System Development

Checkland [1990] Soft Systems Methodology in Action

Avison & Wood-Harper [1990] Multiview

Ved systemudvikling af ikke 100% veldefinerede systemer kan det være nødvendigt/ønskeligt at foretage en nærmere problem analyse, som er snævrere og mere detaljeret end en bred domæne analyse men som stadig går udover selve systemet, der er målet for systemudviklingen.

Mange systemudviklingsmetodologier tager udgangspunkt i et veldefineret system i form af en kravspecifikation og omfatter dermed ikke eller kun i begrænset omfang en egentlig problem analyse. Struktureret systemudvikling omfatter kun simpel problem analyse i form af udarbejdelse af overordnede context og dataflow diagrammer med identifikation af centrale aktører og processer. Objektorienterede systemudviklingsmetodologier omfatter tilsvarende som regel kun simpel problem analyse i form af udarbejdelse af overordnede objekt diagrammer og "use cases" med identifikation af centrale objekter og handlinger. Problem analyse i disse metodologier fokuserer således primært på aspekter der er direkte/objektivt relaterede til eventuel implementering af computerbaserede systemer.

Nogle systemudviklingsmetodologier omfatter dog en egentlig problem analyse. Soft Systems Methodology omfatter detaljeret problem analyse med identifikation og specifikation af organisatoriske, sociale og politiske aspekter i form af "rich pictures", "rod definitioner" og konceptuelle systemmodeller (men omfatter til gengæld ikke direkte eventuel implementering af computerbaserede systemer). Enkelte systemudviklingsmetodologier som for eksempel Multiview integrerer egentlig problem analyse med implementering af computerbaserede systemer i form af en kombination af Soft Systems Methodology og struktureret systemudvikling.

Generel systemudvikling bør ligeledes kunne rumme egentlig, detaljeret problem analyse af organisatoriske, sociale, politiske og eventuelle andre relevevante aspekter samt integreret systemmodellering af disse.

Kravspecifikation

prev next up

Zave & Jackson [1997] Four Dark Corners of Requirements Engineering

Jackson [1995] Software Requirements & Specifications

Målet med et systems kravspecifikation er at beskrive hvad systemet skal opfylde i forhold til omgivelserne uden at beskrive hvordan systemet skal opfylde dette. En kravspecifikation for et system kan dermed defineres som en model af omgivelserne efter (og eventuelt før for at fremhæve forskellene) udvikling og implementering af systemet.

Generelle krav til kravspecifikationer er at de skal være entydige, konsistente og komplette med henblik på umiddelbart at kunne bruges som udgangspunkt for implementering af en løsning samt til verificering af om en implementeret løsning opfylder kravene i kravspecifikationen.

Formulering af en kravspecifikation kompliceres væsentligt af at anvendelsen af den har meget forskellige formål. Kravspecifikationen udgør en (juridisk) kontrakt mellem kunden og systemudviklingsorganisationen, og den definerer løsningen i form af en beskrivelse af grænsefladen mellem omgivelserne og systemet. Derudover udgør kravspecikationen udgangspunktet for kundens forståelse af løsningen og systemudviklernes forståelse af kunden og systemernes omgivelser.

Ideelt set udarbejdes/foreligger der en fuldstændig/tilstrækkelig kravspecifikation som udgangspunkt for det resterende systemudviklingsforløb. I praksis må det dog forventes, at der afdækkes uafklarede krav/problemer i løbet af systemudviklingsforløbet således at kravspecifikationen løbende må tilpasses, samt at nogle specificerede krav eventuelt ikke umiddelbart kan opfyldes og derfor må revurderes og genforhandles.

Et systems kravspecifikation er oftest blot verbale beskrivelser af krav/ønsker til mange forskellige aspekter ved systemet. Dermed er kravspecifikationen meget fleksibel og forholdsvis let at formulere og forstå for såvel "kunden" som systemudviklerne, men det er vanskeligt at sikre entydighed, konsistens og komplethed i kravspecifikationen.

Alternativt kan kravspecifikationen formuleres som en matematisk/logisk baseret formel specifikation, der (automatisk) kan valideres for entydighed, konsistens og komplethed. En formel kravspecifikation er dog forholdsvis svær at formulere og forstå og er således forbeholdt for eksperter, der skal repræsentere såvel "kunden" som systemudviklerne. Formel specifikation mangler desuden udtryksevne og fleksibilitet til at formulere ikke-kvantificerbare krav/ønsker til systemet og må dermed eventuelt suppleres med andre (verbale) beskrivelser.

Til fortrinsvis teknisk orienterede systemer er formel specifikation således velegnet og tilstrækkelig i det omfang at de nødvendige økonomiske, tidsmæssige og eksperticemæssige resourcer er tilgængelige. Ellers må i hvert fald dele af kravspecifikationen bestå af andre (verbale) beskrivelser med den nødvendige forståelighed og fleksibilitet/udtryksevne til også at specificere krav/ønsker til ikke teknisk orienterede aspekter ved systemet/omgivelserne.

Kravspecifikation foregår altså typisk som en separat fase/aktivitet med en separat model uden direkte sammenhæng med den øvrige systemmodellering, hvilket er problematisk i det omfang at løbende tilpasning er nødvendig for at sikre konsistens og sporbarhed mellem kravspecifikationen og den øvrige systemmodellering.

Generel systemudvikling bør således kunne rumme integreret systemmodellering med en kombination af eller en mellemting mellem verbale og formelle specifikationer som sikrer/muliggør den nødvendige/ønskelige fleksibilitet, forståelighed, entydighed, konsistens og komplethed.

Systemanalyse

prev next up

Sommerville [1992] Software Engineering

Yourdon [1989] Modern Structured Analysis

Champeaux, Lea & Faure [1993] Object-Oriented System Development

Systemanalysens mål er forståelse og modellering af henholdsvis de relevante elementer og relationer indenfor systemets domæne samt systemets omgivelser og grænsefladerne til disse som er nødvendige for at designe og implementere systemet. Analysen af de relevante elementer indenfor systemet omfatter data, processer og tilstande/adfærd som udgør de centrale elementer for at designe og implementere det funktionelle system. Analysen af systemets omgivelser og grænseflader omfatter omgivelsernes forudsætninger/begrænsninger samt brugsscenarier og interaktionsmønstre for omgivelsernes brug af/kontakt med systemet.

Den funktionelle systemanalyse i struktureret systemudvikling sker ved hjælp dataflowdiagrammer, der angiver relationer mellem processer i form af dataoverførsel samt processers struktur i form af hierarkisk dekomposition. Datastrukturer angives i form af entitetsrelationsdiagrammer. Med Information Engineering tages der omvendt udgangspunkt i systemets data ligeledes i form af entitetsrelationsdiagrammer og de nødvendige processer for at skabe og anvende data, analyseres og modelleres ud fra disse med hierarkisk dekomposition af processerne. Med objektorienteret systemudvikling sker den tilsvarende funktionelle systemanalyse ved hjælp af objekt strukturdiagrammer der angiver datastrukturer i form af objektklasse hierarkier samt aggregerede objekter og attributter. Objekt interaktionsdiagrammer angiver systemets processer i form af metoder og events for/mellem objekterne.

Analysen af omgivelserne og grænsefladerne til disse er i struktureret systemudvikling og Information Engineering begrænset til en simpel identifikation af eksterne aktører og funktionel specifikation af inddata og uddata mellem aktørerne og systemet. Struktureret systemudvikling og Information Engineering beskæftiger sig således ikke med omgivelsernes kontekst i videre omfang. Objektorienteret systemudvikling omfatter derimod udarbejdelse af "use cases", der er verbale beskrivelser af omgivelsernes kontekst i forbindelse med brug af systemet i form af (sekventielle) interaktionsmønstre mellem omgivelserne og systemet for forskellige handlingsforløb.

Såvel struktureret systemudvikling som Information Engineering og objektorienteret systemudvikling er således meget funktionelt orienterede i analysen af systemets omgivelser og grænseflader. Andre systemudviklingsmetodologier som for eksempel Soft Systems Methodology og Multiview omfatter meget mere varierede og detaljerede analyser af systemets omgivelser i form af blandt andet aktørernes roller og tanker/bekymringer.

Selvom det grundlæggende er de samme elementer i form af data og processer de forskellige systemudviklingsmetodologier modellerer så er der tale om meget forskellige indgangsvinkler til modelleringen som belyser meget forskellige aspekter ved elementerne og systemet, og de enkelte systemudviklingsmetodologier er som regel begrænset til én synsvinkel på systemet.

Ingen af synsvinklerne kan med rette betegnes som værende mere korrekte end andre og et ønske til generel systemudvikling er i den enkelte situation at kunne tage udgangspunkt i den bedste egnede indgangsvinkel som udmærket kan variere for de forskellige dele af et system og for den enkelte systemudvikler. Ligeledes er det ønskeligt løbende at kunne skifte mellem forskellige synsvinkler på det enkelte (del)system for at fremhæve forskellige aspekter. Endvidere må generel systemudvikling også kunne omfatte en bredere og mere varieret analyse end de snævert funktionelt orienterede elementer i form af data og processer, således at også sociale, organisatoriske og politiske aspekter med flere kan inddrages.

Systemdesign

prev next up

Sommerville [1992] Software Engineering

Yourdon [1989] Modern Structured Analysis

Champeaux, Lea & Faure [1993] Object-Oriented System Development

Systemdesign fasens/aktivitetens mål er at specificere en løsning (eller flere) for det pågældende systemudviklingsforløb i form af en model af de relevante elementer og relationer indenfor systemets domæne samt systemets grænseflader til og interaktion med omgivelserne. som eventuelt skal implementeres som et fungerende system. Systemdesignet omfatter således data, processer og tilstande/adfærd som udgør de centrale elementer for at implementere det funktionelle system. Designet af grænsefladerne omfatter eventuel brugergrænseflade design og/eller teknisk interface design samt specifikation af procedurer for brug af systemet og kommunikationsprotokoller.

Systemdesign i struktureret systemudvikling sker med udgangspunkt i systemanalysens modeller i form af dataflowdiagrammer og entitetsrelationsdiagrammer. Dataflowdiagrammer omformes til hierarkiske strukturdiagrammer med angivelse af dataoverførsel mellem forskellige moduler suppleret med specifikation af bearbejdnings- og kontrolprocedurer i struktureret naturligt sprog. Entitetsrelationsdiagrammer "partitioneres" og "normaliseres" med henblik på eventuel databaseimplementering. Information Engineering omfatter tilsvarende systemdesign aktiviteter. Med objektorienteret systemudvikling foregår systemdesign på samme måde som systemanalysen i form af objekt strukturdiagrammer og interaktionsdiagrammer. Objekterne omformes dog fra forretningsorienterede objekter til implementeringsorienterede objekter, som kan være væsentligt forskellige.

Systemdesign af grænsefladerne omfatter i struktureret systemudvikling og Information Engineering design af skærmbilleder/rapporter og relationer mellem disse uden nærmere retningslinier for dette. Objektorienteret systemudvikling omfatter ligeledes design af skærmbilleder/rapporter samt objekter til håndtering af disse og sammenkobling med andre objekter.

Der anvendes således mange og meget forskellige metoder til systemdesign, hvoraf ingen med rette kan siges at være mere korrekte end andre. Systemdesign omfatter grundlæggende de samme elementer i form af data og processer, men de forskellige systemudviklingsmetoder tager udgangspunkt i og fremhæver forskellige aspekter ved elementerne og systemet. Systemudviklingsmetodologier er som regel begrænset til én synsvinkel på systemet.

Fælles for systemdesign metoderne gælder at systemdesign modellerne adskiller sig væsentligt fra de relaterede systemanalyse modeller. En del systemdesign modelleringssprog anvender helt andre koncepter og præsentationsformer end de relaterede systemanalyse modelleringssprog. Selv hvis modelleringssprog koncepterne og præsentationsformerne er indholdet væsentligt anderledes således at der reelt er tale om separate modeller for henholdsvis systemanalyse og systemdesign.

Systemdesign metoderne har desuden begrænset fleksibilitet idet de kun omfatter nogle få programmeringsorienterede beskrivelseselementer. Systemdesign modelleringssprogene er naturligt mere programmeringsorienterede end de relaterede systemanalyse modelleringssprog som traditionelt mellemtrin mellem systemanalyse og implementering. Systemdesign modelleringsprogene afspejler således i vid udstrækning koncepterne fra de anvendte programmeringssprog.

Generel systemmodellering bør således omfatte integreret modellering af det funktionelle system såvel som systemets grænseflader til omgivelserne, og modelleringen af det funktionelle system bør kunne rumme både det computerbaserede system og eventuelle ikke computerbaserede omgivelser. Den generelle systemmodellering bør løbende kunne skiftes mellem forskellige synsvinkler uden at være direkte eller indirekte bundet til bestemte implementeringssprog.

Programmering

prev next up

Finkel [1995] Advanced Programming Language Design

Avison & Fitzgerald [1995] Information Systems Development

Programmerings fasens/aktivitetens mål er at implementere den endelige, eksekverbare systemmodel i form af et eller flere computerbaserede programmer. Programmering sker med mange mere eller mindre forskellige programmeringssprog indenfor forskellige programmeringsparadigmer.

Imperative programmer i programmeringssprog som Pascal og C består af sekvenser af programinstruktioner opdelt i mindre funktioner/procedurer for lettere håndtering og genbrug af programdele. Objektorienterede programmer i programmeringssprog som C++ og Java består ligeledes af sekvenser af programinstruktioner opdelt i mindre metoder grupperet omkring relaterede datastrukturer for lettere håndtering og genbrug af programdele i højere grad end imperative programmer. Andre programmeringssprog som SQL (Structured Query Language) er mindre generelt anvendelige men indeholder standardfunktioner for effektiv udvikling af bestemte typer af programmer.

Højniveau programmeringssprog er abstraktioner af ofte benyttede sekvenser og mønstre af basale maskinekode programinstruktioner og datastrukturer. Forskellige programmeringssprog indenfor de enkelte paradigmer omfatter de samme, fælles koncepter og varierer primært ved forskelle i programmeringssprog syntaksen og detailimplementeringen af koncepterne. Forskellige programmeringsparadigmer som henholdsvis imperative og objektorienterede programmeringssprog omfatter også mange fælles koncepter og forskellen består primært i objektorienterede programmeringssprog er udvidet med formalisering og effektivisering af retningslinier for "gode" programmeringsteknikker som indkapsling af abstrakte datatyper.

I sidste ende er de forskellige programmeringssprog dog blot virtuelle maskiner der alle er opbygget af de samme basale progaminstruktioner. De forskellige programmeringssprog er således blot forskellige abstraktioner og præsentationer for de samme grundlæggende ting og i princippet er det muligt at oversætte mellem dem. Nogle programmeringssprog har dog indbyggede begræsninger således at de ikke er i stand til at udtrykke bestemte, typisk maskinnære, programinstruktioner og datatyper.

I forbindelse med systemudviklingsmetodologier og -værktøjer er der typisk et markant skift i systemmodelleringsprog ved overgangen fra analyse og design til implementering - fra uformel og abstrakt diagrammering til eksakte og konkrete programinstruktioner, og der skelnes ofte mellem Upper CASE og Lower CASE systemudviklingsværktøjer. Ved struktureret systemudvikling fungerer design systemmodelleringen i en vis grad som et mellemsprog mellem de vidt forskellige systemmodelleringssprog i henholdsvis analyse og implementering. Ved objektorienteret systemudvikling anvendes de samme grundlæggende koncepter ved såvel analyse og design som implementering men ved overgangen til implementering tillægges koncepterne konkrete betydninger og begrænsninger. Implementeringen af de enkelte objektmetoder sker desuden i vid udstrækning som imperativ programmering.

Sikring af konsistens mellem analyse/design systemmodeller og implementeringen som programinstruktioner kompliceres af de meget forskellige systemmodeller, og ved at programmeringen er den første egentlige "dialog" med systemet, hvilket eventuelt medfører mange og væsentlige ændringer i forhold til analyse og design systemmodellerne. Der er behov for at programmeringen kun sker indenfor rammerne af analyse/design systemmodellerne, eller at ændringer i forbindelse med programmeringen afspejles i analyse/design systemmodellerne. Ved ændringer af analyse/design er der ligeledes behov for eventuel omstrukturering af programmerne. Kun få, avancerede systemudviklingsværktøjer understøtter gensidig sikring af konsistens mellem analyse/design og programmering, og disse er begrænset bestemte programmeringssprog dialekter og implementeringer.

Da forskellige programmeringssprog er abstraktioner for sekvenser og mønstre af de samme basale elementer, bør de kunne forenes igennem generalisering af koncepterne i et fælles grundlæggende systemmodelleringssprog. Derigennem bør de også kunne integreres bedre og lettere med den øvrige systemmodellering.

Maskinarkitektur

prev up

Tanenbaum [1990] Structured Computer Organization

Tanenbaum [1992] Modern Operating Systems

Finkel [1995] Advanced Programming Language Design

Den klassiske computer maskinarkitektur består af én lagerenhed til lagring af programkode og data samt én processor til afvikling af programmer. Dette har været den almindelige maskinarkitektur igennem årtier. Den bliver dog i stadig stigende grad erstattet med mere komplekse multiprocessor og distribuerede maskinarkitekturer for at opnå den nødvendige kapacitet og for integration af fysisk adskilte ressourcer. Multiprocessor maskinarkitekturer med delt lager består af flere processorer og én fælles lagerenhed samlet i én computer. Distribuerede maskinarkitekturer består af flere computere med hver sine lagerenheder og processorer.

Traditionel systemudvikling har i vid udstrækning udviklet sig med udgangspunkt i den klassiske maskinarkitektur, og den har ikke uden videre kunnet videreføres til de mere komplekse maskinarkitekturer. Imperative programmeringssprog er med én sekvens af programinstruktioner, der bearbejder lagerceller i én lagerenhed, orienteret mod den klassiske maskinarkitektur. For at kunne anvendes til de komplekse maskinarkitekturer er imperative programmeringssprog blevet udvidet med specielle koncepter og funktioner som processer, interproces kommunikation, "threads" og semaforer. Disse er med til væsentligt at øge kompleksiteten i programmer og medfører specielle problemer i form af "deadlock" og "timing" problemer.

Objektorienterede programmeringssprog benytter principielt uafhængige, kommunikerende objekter der fungerer parallelt, og de er demed umiddelbart mere egnede til de komplekse maskinarkitekturer. I praksis fungerer objektorienterede programmeringssprog dog i vid udstrækning som de sekventielle imperative programmeringssprog med et styrende objekter der sekventielt overgiver kontrollen til andre objekter. Disse er således heller ikke umiddelbart velegnede til de komplekse maskinarkitekturer og kræver ligeldes specielle koncepter og funktioner.

Der er behov for at generel systemmodellering omfatter parallelitet som en naturlig del af systemmodelleringen og dermed umiddelbart er velegnet i forbindelse med moderne, distribuerede maskinarkitekturer og deraf følgende programmeringsmodeller. Generel systemmodellering bør naturligvis også være anvendelig til den klassiske, simple maskinarkitektur, men denne kan betragtes som et special tilfælde af distribuerede maskinarkitekturer og parallelitet kan uden større problemer simuleres i simple maskinarkiturer.

Systemudviklingsværktøjer

prev up

Boehm, Penedo, Stuckle, Williams & Pyster [1984] A Software Development Environment

Charette [1987] Software Engineering Environments

Saffo [1996] The Consumer Spectrum

Kuhn [1996] Design for People at Work

Brown & Duguid [1996] Keeping It Simple

Brooks [1995] The Mythical Man-Month

Der findes forskellige systemudviklingsværktøjer til de forskellige systemudviklingsmetodologier, og til de enkelte systemudviklingsmetodologier findes der flere forskellige systemudviklingsværktøjer.

Typisk understøtter systemudviklingsværktøjer kun én systemudviklingsmetodologi eller i hvert fald én familie af tæt relaterede systemudviklingsmetodologier. Typisk understøtter systemudviklingsværktøjer ligeledes kun dele af systemudviklingsmetodologien som for eksempel analyse og design uden implementering. Endeligt understøtter systemudviklingsværktøjer typisk kun et ensrettet systemudviklingsforløb fra analyse over design til implementering. Sådanne begrænsninger i systemudviklingsværktøjer skyldes formodentligt den meget store bredde og kompleksistet der kræves for at understøtte alle faser/aktiviteter i vilkårlig rækkefølge for flere forskellige systemudviklingsmetodologier. Dette vil kræve et systemudviklingsværktøj der understøtter mange, meget forskellige modelleringssprog og oversættelse mellem dem.

Lukkede systemudviklingsværktøjer har begrænsede muligheder for at udvide/tilpasse dem, og det er ikke umiddelbart muligt at udskifte de enkelte dele eller at kombinere dem med andre systemudviklingsværktøjer. Udvidelser og kombineringer af systemudviklingsværktøjer er typisk begrænset til import/eksport og konvertering af data, hvilket vanskeliggøres på grund af manglende standarderformater for data.

Delvist åbne systemudviklingsværktøjer er typisk bundet til én familie af modelleringssprog uden umiddelbar mulighed for at udskifte/kombinere med forskellige sprog. Der er dog muligheder for at udvide/tilpasse systemudviklingsværktøjet med yderligere systemudviklingsværktøjer i form af for eksempel redigeringsværktøjer og præsentationsværktøjer. Udvidelses-/tilpasningsmulighederne er dog typisk begrænset til systemkonfiguration af systemudviklingsværktøjer (i modsætning til fuldt integreret udvidelse/tilpasning af systemudviklingsværktøjets almindelige brugerfunktionalitet) og sikring af data konsistens.

Med komplekse systemudviklingsværktøjer og behov for at anvende forskellige systemudviklingsværktøjer til forskellige systemudviklingsprojekter og -aktiviteter/-faser er det kompliceret og dermed svært og kostbart for systemudviklere at beherske forskellige systemudviklingsværktøjer. Kompleksiteten for anvendelse af forskellige systemudviklingsværktøjer er formodentlig en væsentlig årsag til en vis konservatisme hos systemudviklere og systemudviklingsorganisationer.

Åbne systemudviklingsværktøjer giver derimod et systemudviklingsmiljø med umiddelbare muligheder for at udskifte/kombinere forskellige modelleringssprog igennem et fælles, underliggende (meta)modelleringssprog. Åbne systemudviklingsværktøjer giver ligeledes umiddelbar mulighed for at udvide/tilpasse den almindelige brugerfunktionalitet med direkte integration af udvidelser/tilpasninger. Generel systemmodellering bør kunne udgøre grundlaget for sådanne åbne systemudviklingsværktøjer.

Konklusion på centrale problemer ved systemudvikling

prev

Systemudviklingsfilosofi

Systemopfattelse

Systemudviklingsformål

Systemudviklingsforløb

Systemudviklingsparadig-mer

Systemudviklingsmetodol-ogier

Systemmodellering

Systemudviklingsværktøje-r

Der er et generelt problem for systemudvikling ved at systemudviklingsforløb, -metodologier og -værktøjer bevidst eller ubevidst i vid udstrækning er baseret på en grundlæggende antagelse om at systemudvikling foregår som en ensrettet, rationelt fremadskridende proces, men dette er et uopnåeligt/naivt ideal for mange systemudviklingsopgaver. Dermed vil traditionelle systemudviklingsforløb, -metodologier og -værktøjer komplicere systemudviklingen ved at den løbende skal indpasses i nogle rammer der er grundlæggende forskellige. Der er i stedet for behov for at understøtte mere dialektiske eller direkte kaotiske systemudviklingsprocesser med tæt, gensidigt samspil mellem de forskellige systemudviklingsaktiviteter og ad hoc behov for valg af eller definition af relevante systemudviklingsaktiviteter i forbindelse med de enkelte systemudviklingsforløb og -problemer.

Den grundlæggende systemopfattelse er ligeledes et generelt problem idet systemudviklingsmetodologier, -metoder og -værktøjer typisk kun adresserer én form for system der kun dækker en del af systemudviklingen for eksempel i form af et socialt system eller et teknisk system. Dette komplicerer systemudviklingen ved at hele systemudviklingen skal indpasses i én systemopfattelse der delvist er grundlæggende forskellig, eller ved at systemudviklingen løbende skal koordinere/integrere flere forskellige systemopfattelser og systemmodeller for forskellige dele af systemet. Der er i stedet for behov for en grundlæggende, bredere systemopfattelse, der kan dække hele systemudviklingen.

Formålet med systemudviklingsopgaver varierer meget fra rent teknisk til rent organisatorisk systemudvikling og omfatter typisk flere forskellige grundlæggende formål i forskelligt omfang. Systemudviklingsmetodologier, -metoder og -værktøjer der er orienteret mod ét grundlæggende systemudviklingsformål komplicerer således systemudviklingen ved at hele systemudviklingen skal indpasses i nogle rammer der grundlæggende er uegnede. Der er i stedet for behov for bredere systemudvikling der muliggør såvel den nødvendige detaljering og præcision for udvikling af tekniske systemer som uklarheder, alternativer og konflikter for udvikling af organisatoriske systemer. Systemudviklingen bør ligeledes understøtte meget korte systemudviklings livscykler for hurtig, evolutionær systemudvikling uden at dette nødvendigvis degraderer effektiviteten af det implementerede tekniske system.

Med nødvendigheden af at understøtte mere eller mindre kaotiske systemudviklingsprocesser og forskellige systemudviklingsformål er det ikke muligt at definere faste aktiviteter/faser og (mellem)produkter for systemudviklingsforløb. Der er behov for at definere/tilpasse disse i forbindelse med det enkelte systemudviklingsforløb.

Den typiske skelnen mellem forskellige systemudviklingsparadigmer med udgangspunkt i forskellige synsvinkler som henholdsvis data, processer og objekter samt analytisk top-down nedbrydning og konstruktivistisk bottom-up opbygning er ikke hensigtsmæssig. De forskellige synsvinkler er alle relevante og nyttige i forbindelse med systemudvikling. Der er således behov for at disse "paradigmer" netop gøres til synsvinkler, der frit kan anvendes i systemudviklingen som forskellige måder at præsentere systemet på.

Den samme valgfrihed gælder for valg af og løbende skift mellem forskellige systemudviklingsmetodologier og -metoder tilknyttet de forskellige systemudviklingsparadigmer for at kunne udnytte de forskellige synsvinkler fuldt ud. Der er behov for at generalisere de forskellige systemudviklingsmetoder eller i hvert fald de systemmodeller de bearbejder således at de frit kan kombineres i stedet for at systemudviklingsmetoder er bundet til hinanden enkeltvis i form af specielle systemmodeller som henholdsvis inddata og uddata.

For frit at kunne kombinere forskellige systemudviklingsmetoder er det nødvendigt at generalisere systemmodellerne de bearbejder, da det ellers medfører nærmest uendelig kompleksitet i bindingerne mellem de forskellige systemudviklingsmetoder og -modeller på

grund af et stort og eksponentielt stigende antal bindinger mellem de enkelte systemudviklingsmetoder og -modeller. Der er således behov for at generalisere de forskellige systemmodeller til én komplet og konsistent systemmodel med fælles, grundlæggende koncepter og elementer. Ellers må systemudvikling begrænses til et meget begrænset antal systemudviklingsmetoder og -modeller, da kombinationen af disse ellers vil være for kompleks for håndtering af systemudviklerne og systemudviklingsværktøjerne.

Med behov for at benytte forskellige systemudviklingsmetodologier og -metoder er det urealistisk/upraktisk at disse alle skal være omfattet af et enkelt systemudviklingsværktøj, og der kan desuden være behov for at benytte ad hoc og eksperimentielle systemudviklingsværktøjer. Der er dermed behov for åbne systemudviklingsværktøjer der frit kan kombineres/integreres, hvilket ligeledes kræver anvendelse af en generaliseret systemmodel som fælles grundlag for de forskellige systemudviklingsværktøjer.

Interessante idéer og udviklinger indenfor systemudvikling

next up level

Integreret systemmodellering

Formalisering

Flow orientering

I det følgende analyseres en række forskellige idéer og udviklinger indenfor systemudvikling. Der er primært tale om nyere "state-of-the-art" samt ældre idéer og udviklinger der endnu ikke er fuldt udviklede og har fundet bred anvendelse.

Med den stadig øgede udbredelse og standardisering af objektorienteret systemudvikling omfattende alt fra systemanalyse til implementering og distribueret maskinarkitektur anvendes der i stadig stigende grad forenede systemmodellering koncepter, metoder og værktøjer for bedre integreret og mere effektiv systemudvikling.

Anvendelsen af formelle metoder i systemudvikling har kun langsomt fundet begrænset anvendelse til systemudvikling, men med øget brugervenlighed og tilgængelighed for formelle metoder igennem integration med traditionel systemudvikling samt bredere, højniveau simulering modelleringssprog og - metoder er der grundlag for bredere anvendelse af formelle metoder i systemudvikling. Derudover er indholdsmæssig, højniveau formalisering i form af problem, analyse og design mønstre i hastig udvikling og udbredelse.

Endeligt er systemudvikling i stigende grad flow orienteret med distribuerede maskinarkitekturer, flow programmering samt slutbruger konfigurerbare applikationer til afløsning for simple maskinarkitekturer, sekventielle programmeringsmodeller og lukkede, enkeltstående applikationer.

Integreret systemmodellering

next up down

Objektorientering

Unified Modelling Language

Meta Object Facility

CORBA

Med udbredelsen af fælles objektorienterede koncepter fra objektorienterede programmeringssprog til objektorienteret analyse og design har systemudvikling fået et forenende fundament for en stor del af hele systemudviklingsforløbet. Det bør være med til at gøre systemudviklingen lettere og mere konsistent for systemudviklerne fremfor at skulle skifte mellem vidt forskellige koncepter som for eksempel dataflowdiagrammer, strukturdiagrammer og imperativ programmering i forbindelse med struktureret systemudvikling.

Objektorienteret systemudvikling er ligeledes blevet forenet igennem et standardiseret objektorienteret modelleringssprog, der giver systemudviklerne fælles syntaktiske og semantiske retningslinier for objektorienterede systembeskrivelser på tværs af systemudviklingsprojekter, -organisationer og -værktøjer. Dermed skulle kommunikation mellem systemudviklere og overdragelse af systemer blive lettere idet man kan fokusere på selve indholdet fremfor varierende og eventuelt forvirrende præsentationer.

Modelleringssproget er desuden underbygget med et meta-modelleringssprog og implementering af dette som en facilitet, som systemudviklingsværktøjer kan baseres på, således at forskellige, uafhængige systemudviklingsværktøjer kan kombineres og substitueres med hinanden. Dermed øges mulighederne for at vælge det "rigtige" værktøj til den enkelte opgave, og der skabes øget konkurrence om nye og bedre systemudviklingsværktøjer og funktioner i disse.

Endeligt understøttes modelleringssproget af standarder for (specifikation af) implementering af objektorienterede programmer/komponenter samt en arkitektur for eksekvering af programmer/komponenter på tværs af forskellige programmeringssprog, styresystemer og hardware platforme.

Det kan diskuteres om de objektorienterede koncepter, UML (Unified Modelling Language), MOF (Meta Object Facility) og CORBA (Common Object Request Broker Architecture) er de rigtige løsninger eller om det kan gøres bedre på andre måder. Hvis det kan gøres bedre, kan det desuden diskuteres om den objektorienterede standardisering i praksis vil begrænse udviklingen og udbredelsen af andre løsninger. Principielt må forenede/forenende koncepter og implementeringer dog ses som en væsentlig forbedring af systemudvikling i form af enklere og bedre integrerede systemudviklingsforløb.

Objektorientering

next up

Wirth [1976] Algorithms & Data Structures = Programs

Kim & Lochovsky [1989] Object-Oriented Concepts, Databases, and Applications

Züllighoven [1998] The Tools & Materials Approach

Med objektorienteret systemudviklings start som programmeringssprog introduceredes de grundlæggende objektorienterede koncepter som senere også har fundet anvendelse i forbindelse med system analyse og design.

Objektorienteret programmering kan ses som en videreudvikling og formalisering af struktureret progammerings abstrakte data typer i form af objekter som selvstændige enheder der omfatter såvel data/datatyper som processer. Objekter giver muligheder for bedre opdeling af programmer i mindre enheder og gruppering af relaterede data/datatyper og processer.

Bindingerne mellem forskellige programdele mindskes igennem specifikation af objekternes eksterne interface og indkapsling af implementeringsdetaljer og mindsker dermed kompleksiteten i det samlede system. Endeligt giver objektklasser og nedarvning af egenskaber fra disse mulighed for samlet beskrivelse af fælles, generelle egenskaber ved forskellige "ting"/fænomener samt specialisering af disses særlige egenskaber. Dette kan ligeledes medvirke til at mindske kompleksiteten i det samlede system.

(Anvendelse af objektorienterede koncepter i programmeringssprog er dog ikke nogen garanti for mindsket/bedre håndtering af kompleksistet i systemet. Dårlig implementering af objektorienterede koncepter i de enkelte progammeringssprog og anvendelse af disse kan tværtimod medvirke til væsentlig øget kompleksitet. For eksempel i form af løst specificerede eksterne interfaces og "overriding" af nedarvede metoder der kan resultere i uigennemskuelige bindinger mellem forskellige objektklasser.)

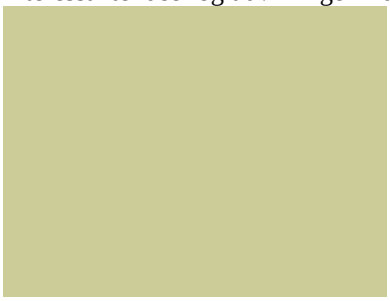
De samme objekterorienterede koncepter anvendes ligeledes i forbindelse med analyse og design og produkterne af disse aktiviteter er ligeledes objekt modeller med specifikation af objekter og disses egenskaber i form af "attributter" og "metoder". Der er således ingen principiel/grundlæggende forskel på objektorienteret analyse, design og programmering. Det hele handler om objekter.

De forskellige objektorienterede aktiviteter fokuserer blot på forskellige aspekter ved de i princippet samme objekter. Analyse identificerer objekter og disses egenskaber samt relationer mellem objekterne. Design strukturerer objekterne i objektklasse hierarkier og specifierer objekternes eksterne grænseflader. Programmering implementerer objekterne som konkrete datatyper/-strukturer og algoritmer/fuktioner/procedurer. De forskellige aktiviteter er således ideelt set blot trinvis forfining af de samme objekter.

I praksis holder enhedsmodelleringen igennem objektorienteret analyse, design og programmering dog ikke helt, idet de enkelte aktiviter eventuelt må arbejde med separate og måske meget forskellige objekt modeller. Objekterne ved analyse er fænomener/begreber i "den virkelige verden" - eventuelt med uformaliserede egenskaber. Objekterne ved design er computerrelaterede konstruktioner - eventuelt uden direkte relation til fænomener/begreber i "den virkelige verden". Objekterne ved programmering er konkrete implementeringer udfra bestemte programmeringssprogs konkrete måske begrænsede implementering af de objektorienterede koncepter - eventuelt uden direkte en-til-en relation til de abstrakte design konstruktioner.

Muligheden for objektorienteret systemudvikling som et tæt gensidigt sammenspil mellem analyse, design og programmering begrænses dermed, i det omfang der arbejdes med forskellige modeller hvis elementer ikke er direkte relaterede, da det kræver (manuel) opdatering af de forskellige modeller for at sikre konsistens imellem dem. Dette trækker således i retning af et traditionelt, ensrettet systemudviklingsforløb.

En vigtig kvalitet ved objektorientering er at objekter hævdes at være "naturlige" i forhold til "den virkelige verden" for såvel brugere/kunder som systemudviklere. Dette kan dog diskuteres, og det er i hvert fald ikke altid tilfældet. For eksempel er det ikke umiddelbart "naturligt" på hvilke(t) objekt(er) metoder skal placeres i forbindelse med processer der involverer flere, ligeværdige objekter uden ét objekt der er naturligt styrende. I forbindelse



med objektorienteret analyse er identificeringen af objekter heller ikke altid umiddelbart naturlig, og der er givet meget forskellige, mere konkrete forslag til identificering af de "rigtige" objekter som for eksempel fysiske artifakter i "den virkelige verden" og problemmønstre.

På trods af disse problemer er objektorienterings forenende koncepter interessante, og problemerne er ikke ældre og større end at der stadig er håb om løsninger/forbedringer.

Unified Modelling Language

prev next up

Object Managerment Group [1997] Unified Modelling Language

Unified Modelling Language (UML) er en standard i regi af brancheorganisationen Object Management Group, som specificerer semantik og notation for et objektorienteret modelleringssprog. Formålet med standardiseringen er at lave et umiddelbart anvendeligt, visuelt objektorienteret modelleringssprog med en formel basis og med indbyggede udvidelsesmuligheder, som kan være fundament for udvikling af objektorienterede systemudviklingsværktøjer, der dermed lettere kan udvikles, anvendes, kombineres og integereres. Implementering af UML understøttes desuden af Meta Object Facility (MOF) standarden.

Standardiseringen sker som en reaktion mod fragmenteringen indenfor objektorienterede systemudviklingsmetodologier, -metoder og -værktøjer, som omfatter ensartede men alligevel mere eller mindre forskellige og løst formulerede objektorienterede koncepter og modelleringssprog. Det standardiserede modelleringssprog er lavet med udgangspunkt i tre af de førende, mest udbredte objektorienterede systemudviklingsmetodologiers modelleringssprog af Booch, Rumbaugh og Jacobson.

UML standarden omfatter en række pakker der hver især specificerer forskellige objektorienterede koncepter. De primære pakker specificerer en basis objektmodel med centrale objektorienterede koncepter som objekter, klasser, nedarvning og specialisering samt basis data typer for attributter. Derudover specificerer sekundære pakker en række forskellige objektorienterede koncepter som for eksempel Use Case og State Machine. For at UML også skal kunne rumme ikke-standardiserede koncepter (selvudviklede eller fra andre systemudviklingsmetodologier) angiver en primær Extension pakke desuden udvidelsesmuligheder af modelleringssproget i form af pseudo klasser og attributter, som kan bruges til at definere nye modelelementer.

UML bærer præg af at være lavet med udgangspunkt i bestemte objektorienterede systemudviklingsmetodologier og programmeringssprog i form af de valgte modelelementer. Andre modellerings- og implementeringsalternativer som for eksempel data-flow diagrammer og ikke-"message passing" baserede programmeringssprog er udelukkede eller henvist til modellering som pseudo klasser og attributter uden egentlig egen semantik og notation, og hvis understøttelse i standard systemudviklingsværktøjer er meget begrænset. De understøttes eventuelt kun som rene diagrammeringselementer uden betydning ved for eksempel diagramvalidering og programkodegenerering. Mere omfattende understøttelse af udvidelser med egen semantik og notation ér dog identificeret som en mangel og bliver eventuelt medtaget i en senere version af UML.

Selvom UML er forsøgt begrænset til de centrale koncepter, er standarden dog rimeligt omfattende og kompleks, og gennemlæsning af specifikationerne efterlader ikke umiddelbart læseren med et klart indtryk af modellering med UML. UML specifikationerne er dog heller ikke rettet mod "almindelige" brugere, men omfanget og kompleksiteten af disse indikerer dog alligevel, hvor meget den "almindelige" bruger bevidst eller ubevidst skal kunne håndtere i forbindelse med UML modellering.

Med en formel modelleringssprog standard er der mange fordele for systemudvikling i form af fælles sprog og værktøjer som fundament for metodologier og metoder, men der er en risiko for at standardiseringen bremser for videre udvikling i det omfang at standarden ikke er "rigtig" og tilstrækkelig. Forhåbentligt vil UML kunne videreudvikles i takt med nye idéer, og idéen om et standardiseret fundament er dog i hvert fald interessant, og alternativet med mange forskellige, uforenelige metodologier, metoder, modelleringssprog og værktøjer er ikke attraktivt.

Meta Object Facility

prev next up

Object Management Group [1997] Meta Object Facility

Meta Object Facility (MOF) er også en standard i regi af brancheorganisationen Object Management Group, som specificerer en meta-metaobjektmodel semantik og grænseflade for repræsentering og implementering af metamodeller som for eksempel Unified Modelling Language (UML). Formålet med standardiseringen er at specificere faciliteter til understøttelse af implementering af for eksempel systemudviklingsværktøjer og systemudvikling "repositories" i distribuerede miljøer.

MOF er specificeret i relation til UML (men ikke kun med henblik på at understøtte UML) og omvendt. MOF og UML er derfor ensartet opbygget og specificerer mange af de samme koncepter uden at der dog er en en-til-en sammenhæng mellem dem. MOF er desuden specificeret i relation til Common Object Request Broker Architecture (CORBA) standarden med henblik på implementering af MOF som CORBA komponenter/services.

MOF specifikationen er dermed også forholdsvist omfattende og kompleks, da den skal omfatte alle koncepterne i UML (samt relaterede systemudviklingsværktøjer og (meta)model "repositories") og binde dem sammen med en "gammel" standard som CORBA. Som generel meta-metamodel er den meget omfattende og kompleks, og den omfatter en del objektorientering og programmerings specifikke koncepter som eventuelt ikke er generelt anvendelige/nødvendige.

Specifikationen muliggør dog implementering af standard værktøjer til understøttelse af UML og andre (meta)model relaterede applikationer, og den er således i hvert fald interessent indtil at der eventuelt bliver lavet bedre alternativer. MOF specifikationen er desuden heller ikke endeligt afsluttet, og der foregår stadig videreudvikling af den.

CORBA

prev up

Object Management Group [1997,1998] CORBA

Common Object Request Broker Architecture (CORBA) er yderligere en standard i regi af brancheorganisationen Object Management Group, som specificerer definition og implementering af objektorienterede komponenter og applikationer. Formålet med standardiseringen er at specificere programgrænseflader og faciliteter således at forskellige komponenter og applikationer kan kommunikere med hinanden på tværs af netværk, hardware, styresystem og programmeringssprog.

CORBA understøtter således genbrug af distribuerede, eksekverbare programdele og giver dermed et væsentligt supplement til traditionel genbrug af lokale inkluderbare/kompilerbare programkildetekstbiblioteker og linkbare funktionsbiblioteker. CORBA giver desuden mulighed for at give "gamle" og eventuelt ikke-objektorienterede komponenter og applikationer en objektorienteret, distribueret grænseflade således at de uden videre kan indgå i CORBAs Object Management Architecture (OMA) for objektorienterede, distribuerede komponenter og applikationer.

Formalisering

prev next up down

Specifikationer

Simulering

Mønstre

Formel specifikation har stort potentiale for at forbedre systemudvikling og kvaliteten af de udviklede systemer ved at kunne sikre eksakte, konsistente og komplette specifikationer af alt fra kravspecifikationer til eksekverbare programmer. Formel specifikation har dog ikke nået udbredt anvendelse i systemudvikling på grund af omfanget og kompleksiteten ved at lave og læse dem for systemudviklerne og kunderne/brugerne.

Ved at kombinere/integrere formel specifikation med uformelle specifikationer som for eksempel dataflow diagrammer i struktureret systemudvikling bliver det meget lettere at udarbejde og læse formelle specifikationer igennem forholdsvis intuitive, simple, visuelle diagrammer i stedet for matematisk/logisk baserede udtryk.

Tilsvarende er formelle sprog, metoder og værktøjer til modellering af simulerbare/eksekverbare modeller som for eksempel Petrinet udviklet fra kun at være praktisk anvendelige til små systemer med simple koncepter til også at være praktisk anvendelige til store systemer med komplicerede koncepter, som er nødvendige i forbindelse med analyse og design af systemer.

Formel specifikation er dermed på vej til at kunne nå udbredt anvendelse i systemudvikling igennem integration i almindelige systemudviklingsværktøjer. Med mulighed for formel specifikation som en integreret del af såvel kravspecifikation, analyse, design og implementering vil de forskellige dele af systemudvikling desuden kunne have ét fælles, underliggende modelleringssprog, som kan sikre konsistens imellem de forskellige dele.

En anden form for formalisering er identificering og beskrivelse af tilbagevendende/gentagne mønstre i problemer, analyse og design samt erfarede hensigtsmæssige løsninger for disse. Med mere eller mindre formaliserede mønstre etableres der et mere specialiseret, fælles sprog for indholdet af systemmodeller udover de traditionelle, fælles sprog for formen af systemmodeller.

Specifikationer

next up

Clarke & Wing [1996] Formal Methods: State of the Art and Future Directions

Fraser, Kumar & Vaishnavi [1991] Informal and Formal Requirements Specification Languages: Bridging the Gap

Baresi & Pezzé [1998] Toward Formalizing Structured Analysis

Larsen, Katwijk, Plat, Pronk & Toetenel [1992] SVDM: An integrated combination of SA and VDM

Plat, Larsen & Toetenel [1993] Formal Transformations: Using SA and VDM as Different Views in Software Development

Larsen, Plat & Toetenel (1993) A Formal Semantics of Data Flow Diagrams

France [1992] Semantically Extended Data Flow Diagrams: A Formal Specification Tool

Agerholm & Larsen (1998) A Lightweight Approach to Formal Methods

På grund af formelle specifikationers omfang og kompleksitet er de ikke umiddelbart anvendelige for almindelige systemudviklere og kunder/brugere, og de kræver derfor involvering af yderligere eksperter/ekspertice, hvilket umiddelbart komplicerer og fordyrer systemudviklingsforløbet.

Da formelle specifikationer ikke er umiddelbart anvendelige for almindelige systemudviklere og kunder/brugere vil de desuden ikke kunne stå alene men skulle suppleres med almindelige systemudviklingsspecifikationer. Dermed bliver der endnu flere forskellige modeller som skal udarbejdes og vedligeholdes for at sikre konsistens i modellerne igennem systemudviklingsforløbet.

Det er dermed hensigtsmæssigt at forsøge at integrere uformelle og formelle specifikationer med det formål at gøre formelle specifikationer lettere at anvende og mere udbredte, samt at styrke de uformelle specifikationers værdi ved at præcisere deres mening/betydning.

Det er ikke umiddelbart muligt at integrere formelle og uformelle specifikationer. For at opnå dette er det nødvendigt at lave en semantisk specifikation for de uformelle specifikationer, og i den forbindelse viser der sig eventuelt mangler og uklarheder i betydningen af de uformelle specifikationer som skal udbedres/afklares. Efter formuleringen af en entydig semantik for de uformelle specifikationer er det muligt at formulere at mapping mellem de uformelle og formelle specifikationer og derigennem automatisk konvertere mellem de forskellige specifikationer. En sådan letvægtsform for formel specifikation giver umiddelbar, let adgang til at udnytte mange af fordelene ved formel specifikation.

Ved integration af formelle og uformelle specifikationer får systemudviklerne mulighed for at modellere systemet med forholdsvist intuitivt letforståelige og letanvendelige uformelle specifikationer som for eksempel dataflow diagrammer som samtidigt har bagvedliggende formelle specifikationer med mulighed for validering og anden matematisk/logisk analyse og bearbejdning som for eksempel simulering.

Simulering

prev next up

Jensen [1997] A Brief Introduction to Coloured Petri Nets

Jensen [1994] An Introduction to the Theoretical Aspects of Coloured Petri Nets

Jensen [1996] An Introduction to the Practical Use of Coloured Petri Nets

Oberweis & Sander [1996] Information System Behavior Specification by High-Level Petri Nets

I forbindelse med systemudvikling er muligheden for at simulere systemet i en model førend den egentlige implementering værdifuld ved på et tidligt tidspunkt at kunne afsløre væsentlige fejl og mangler i det designede system og derved spare tid og resourcer.

Petri net er et sprog/værktøj som er udviklet med en formel basis til modellering og simulering af systemprocesser. Petri net omfattede oprindeligt kun simple ikke-hierarkiske netværk med boolske/binære data, som ikke er særligt anvendelige til store, komplekse systemer med komplekse datastrukturer. Petri net er dog senere udvidet med hierarkiske netværk til opdeling af komplekse systemer; komplekse datatyper som tekst og tal til modellering af almindelige data; og hierarkisk relationelle datastrukturer til modellering af komplekse datastrukturer.

Dermed er Petri net udvidet med funktioner til modellering af store og komplekse systemer svarende til mulighederne med for eksempel dataflow diagrammer. Med Petri net er der desuden mulighed for at anvende modellen til simulering af systemet, samt validering og anden matematisk/logisk analyse og bearbejdning af modellen som følge af den formelle basis.

Mønstre

prev up

Knuth [1973] The Art of Computer Programming, Vol. 1: Fundamental Algorithms

Ledbetter & Cox [1985] Software-ICs

Bassett [1987] Frame-Based Software Engineering

Lanergan & Grasso [1984] Software Engineering with Reusable Designs and Code

Gamma, Helm, Johnson & Vlissides [1994] Design Patterns

Fowler [1997] Analysis Patterns

Jackson [1995] Software Requirements & Specifications

Brown, Malveau, McCormick & Mowbray [1998] Anti Patterns

Standard algoritmer og datastrukturer er almindeligt anvendte værktøjer i forbindelse med systemudvikling i form af funktions-/objektbiblioteker med standard løsninger på standard programmeringsproblemer som for eksempel sorteringsalgoritmer. Relaterede funktioner/objekter er desuden samlet i frameworks, der kan anvendes som fundament for programmeringsløsninger i forbindelse med forskellige, specifikke problemdomæner som for eksempel faktureringssystemer.

Udover at bidrage med genbrugskomponenter til systemudviklingen, bidrager standard algoritmer og datastrukturer også med standard begreber og sprogbrug, som giver systemudviklerne bedre muligheder for umiddelbart at kommunikere i forbindelse med programmeringsløsninger.

Tilsvarende ideer om standard begreber og genbrug af programdesign i forbindelse med forskellige standard programtyper som for eksempel validerings- og rapporteringsprogrammer har eksisteret længe men de har først fundet større udbredelse igennem de senere år.

I forbindelse med objektorienteret systemdesign har designmønstre givet standard begreber og genbrug et gennembrud for andet end programmeringsrelaterede algoritmer og datastrukturer. Designmønstre har startet en hel bevægelse, som desuden også har bredt sig til mønstre og anti-mønstre for systemanalyse, systemarkitektur og problemdomæner med mere.

Designmønstre beskriver tilbagevendende standard designproblemer og gennemprøvede designløsninger samt deres konsekvenser for systemet. Designmønstre er i vid udstrækning verbale beskrivelser samt mere formelle beskrivelser af designløsningerne med strukturbeskrivelse i form af løsningernes elementer/objekter og disses relationer samt (pseudo)programkode til beskrivelse af deres implementering.

Analysemønstre beskriver tilsvarende tilbagevendende standard analyseproblemer og gennemprøvede analyseløsninger for modellering af komplekse (del)systemers elementer og relationer. Analysemønstre er ligeledes verbale beskrivelser samt mere formelle beskrivelser i form af strukturbeskrivelser af modellernes elementer/objekter og disses relationer.

Problemrammer er simple strukturbeskrivelser af enkelte aspekter af et problem i form af nogle få elementer og deres relationer. Problemrammer anvendes til at lave en struktureret opdeling af komplekse problemer i mindre, simplere og delvist veldefinerede/formaliserede delproblemer, hvorfra der kan udledes velegnede problemløsningsmetoder.

Med mønstre er der etableret et væsentligt bedre grundlag for forbedre systemudviklingprocessen idet systemudviklerne derigennem i videre udstrækning har fået et fælles sprog til at kommunikere og fastholde beskrivelser af tilbagevendende problemer og løsninger i forbindelse med systemudvikling.

Med et mere formaliseret begrebsapparat er der ligeledes bedre muligheder for at strukturere og integrere systemudviklingsforløbet. I modsætning til tidligere systemanalyse og systemdesign metodologier/-metoder, som blot foreskriver "naturlig" nedbrydning af systemer til delsystemer, beskriver mønstre relevante delsystemer, og relevante problem-, analyse- og designmønstre samt programkode kan desuden kædes sammen som udgangspunkter og genbrugskomponenter igennem systemudviklingsforløbet.

Flow orientering

prev up down

Distribuerede computere

Flow programmering

Slutbruger konfigurerbare applikationer

Med udviklingen af stadigt større og mere komplekse systemer bliver systemerne i stigende grad udviklet som uafhængige, samarbejdende delsystemer - dels som en nødvendighed ved at nogle delsystemer er fysisk separerede, dels som en måde at håndtere de stigende krav til kapacitet, kompleksitet og størrelse.

Computerbaserede systemer består i stigende omfang af separate computere der kommunikerer over netværk for at skalere til meget stor kapacitet og for at integrere data og resourcer fra forskellige fysiske lokationer. Dette gælder for såvel globale systemer over netværk som Internet, organisatoriske systemer over lokalnetværk og indenfor enkeltstående computere opbygget af flere CPU'er og lagerenheder.

Programmering foregår i stigende omfang som implementering af separate, kommunikerende komponenter for at afspejle og udnytte de distribuerede computer hardware arkitekturer og for at håndtere kompleksiteten og udviklingen af store systemer.

Med stadig stigende omfang af anvendelsen af computerbaserede systemer er der behov for at brugerne (eller "sælgere" tæt på brugerne) kan integrere forskellige mere eller mindre generelle systemer til specifikke, individuelle løsninger som passer til de enkelte, konkrete opgaver.

Distribuerede computere

next up

Tanenbaum [1990] Structured Computer Organization

Tanenbaum [1992] Modern Operating Systems

Den klassiske, von Neuman computer maskinarkitektur består af en processor og en adresserbar lagerenhed til lagring af programmer og data, hvilket har været og stadig er tilstrækkelig til mange computerbaserede løsninger. Med denne simple maskinarkitektur er en programmeringsmodel med programmer der bearbejder data og eventuelt programmer i et adresserbart lagerområde umiddelbart oplagt, enkel og fleksibel, og dette har været den dominerende programmeringsmodel.

Med øgede krav til processorkapacitet, hvor en enkelt, tilstrækkelig kraftig processor ikke umiddelbart er tilgængelig og eventuelt slet ikke er fysisk mulig, er det nødvendigt med en mere avanceret maskinarkitektur med flere processorer. Med denne maskinarkitektur opstår der problemer med den klassiske programmeringsmodel i forbindelse med timing problemer mellem programmerne på de forskellige processorer i forbindelse med deling af lagerområdet. Optimal udnyttelse af maskinarkitekturen komplicerer desuden programmeringsmodellen yderligere.

Med tilsvarende øgede krav til lagerkapacitet, hvor en enkelt lagerenhed ikke er tilstrækkelig på grund af transmissionsbegrænsninger mellem lagerenheden og et antal processorer, er en yderligere avanceret maskinarkitektur med flere lagerenheder nødvendig. Denne maskinarkitektur kræver kompliceret, resourcekrævende simulering af en enkelt lagerenhed eller en udvidet programmeringsmodel med eksplicit transmission af data mellem programmerne på de forskellige processorer. Optimal udnyttelse af maskinarkitekturen komplicerer ligeledes programmeringsmodellen yderligere.

Ved massivt parallelle maskinarkitekturer med mange processorer og lagerenheder samt geografisk adskilte computere, der skal fungere som ét distribueret system, er simulering af en enkelt lagerenhed ikke praktisk mulig, og den udvidede programmeringsmodel med eksplicit transmission af data mellem programmerne på de forskellige computere er nødvendig. Dette bliver iøvrigt det centrale i programmeringsmodellen, og den klassiske programmeringsmodel er kun et lokalt fænomen i den enkelte computer. Det kan derfor ses som en nødvendig, fundamentalt anderledes programmeringsmodel i forhold til den klassiske.

Andre avancerede maskinarkitekturer bryder på tilsvarende måde fundamentalt med den klassiske ved at fjerne/minimere den adresserbare lagerenhed til data således at data udelukkende/primært eksisterer og bearbejdes som inddata og uddata, der transmitteres mellem processorerne (eventuelt også kun en enkelt processor).

Selvom den klassiske maskinarkitektur og programmeringsmodel har været dominerende, er en fundamentalt anderledes maskinarkitektur og programmeringsmodel nødvendig i forbindelse med store, avancerede computersystemer. Den fundamentalt anderledes maskinarkitektur og programmeringsmodel er desuden også mulig og eventuelt hensigtsmæssig i forbindelse med enkeltstående computere og mindre, simplere computersystemer.

Flow programmering

prev next up

Finkel [1995] Advanced Programming Language Design

Morrison [1994] Flow-Based Programming

De dominerende programmeringssprog er naturligvis tæt knyttet til den dominerende klassiske maskinarkitektur og programmeringsmodel med processor der bearbejder data i en adresserbar lagerenhed. Der har dog været en løbende udvikling i programmeringssprog af faciliteter til bedre håndtering af maskinarkitekturen og programmeringsmodellen samt kompleksiteten i de udviklede programmer. Programmeringssprog er ligeledes udviklet løbende i relation til udviklingen fra den klassiske til den distribuerede maskinarkitektur og programmeringsmodel.

Højniveau programmeringssprog som pascal og C begrænser kompleksiteten i programmer ved at begrænse bieffekter af programinstruktioner ved delvist at opdele lagerenheden i mindre, separate lagerområder i form af lokale variable i de enkelte funktioner. Internt i funktionerne fungerer de efter den klassiske programmeringsmodel.

Med abstrakte datatyper og objektorienterede programmeringssprog mindskes kompleksiteten i form af bieffekter af programinstruktioner yderligere ved at begrænse disse til definerede operationer på data. Internt fungerer abstrakte datatyper og objekter dog også efter den klassiske programmeringsmodel.

Den klassiske maskinarkitektur og programmeringsmodel omfatter én processor som gennemløber et program, hvilket også afspejler sig i de dominerende højniveau programmeringssprog som de imperative programmeringssprog og objektorienterede programmeringssprog med passive objekter. I sådanne programmer er én procedure eller ét objekt som udgangspunkt styrende, hvorfra kontrollen trinvis i ordnet rækkefølge overdrages til andre procedurer og objekter.

Imperative programmeringssprog med parallelprogrammeringsudvidelser og objektorienterede programmeringssprog med aktive objekter bryder med den klassiske programmeringsmodel ved at tillade flere forskellige kontrollerende procedurer og objekter indenfor et program for at udnytte multiprocessor maskinarkitekturer. Det kan desuden mindske kompleksiteten af programmer og udviklingen af programmer ved at opdele dem i mindre, samarbejdende enheder, men det øger dog også kompleksiteten med timing og allokering problemer i det omfang at enhederne deler data og ressourcer. Dette vil som regel være tilfældet, da programmeringssprogene grundlæggende, indenfor de enkelte enheder fortsat fungerer efter den klassiske programmeringsmodel.

Væsentlige problemer i forbindelse med de dominerende imperative og objekorienterede programmeringssprog kan henføres til den klassiske programmeringsmodel i forbindelse med bieffekter af programinstruktioners operationer på data i lagerenheden.

Funktionelle programmeringssprog undgår problemer med bieffekter ved at bryde med den klassiske programmeringsmodel ved ikke at adressere og bearbejde data i lagerenhed men i stedet for kun at aflæse og anvende data i lagerenheden som inddata til en funktion, der allokerer og genererer nye data i lagerenheden som uddata. Funktionel programmering virker umiddelbart dog ikke så effektivt med hensyn til såvel udvikling som eksekvering af programmer, og mange programtyper virker umiddelbart ikke så naturligt som funktioner.

Dataflow programmeringssprog undgår ligeledes problemer med bieffekter ved at data kun eksisterer som selvstændige, ikke-adresserbare enheder der flyder/transmitteres som inddata og uddata igennem netværk af programinstruktioner (der alle principielt altid er aktive). Dataflow programmering er funktionel men omfatter mere avancerede data og kontrolstrukturer og virker umiddelbart mere naturligt.

Der kan således ses en udvikling i programmering fra at omfatte én programproces og ét lagerområder med udgangspunkt i den klassiske maskinarkitektur og programmeringsmodel til at omfatte mange selvstændige programprocesser og data (uanset om den underliggende maskinarkitektur er klassisk eller distribueret).

Slutbruger konfigurerbare applikationer

prev up

Object Management Group [1997,1998] CORBA

Object Management Group [1998] CORBA Component Scripting - Revised Joint Submission

Ciancarini [1996] Coordination Models and Languages as Software Integrators

Gelernter & Carriero [1992] Coordination Languages and their Significance

Morrison [1994] Flow-Based Programming

Goldman, Anderson & Swaminathan [1994] The Programmers' Playground

McCartney [1996] End-User Construction and Configuration of Distributed Multimedia Applications

Applikationer har udviklet sig fra at være enkeltstående programmer, der er udviklet fra bunden af, til i stadig stigende grad at være sammensat af eksisterende programdele. Dette er nødvendigt og hensigtsmæssigt for at kunne lave store, komplekse, individuelle løsninger.

Programmer består i vid udstrækning af genbrugsprogramkode i form af funktionsbiblioteker og programkode kildetekst. Disse ting er tæt bundet til det enkelte program og er primært under programmørens kontrol.

Igennem de senere år har objektorienterede softwarekomponent arkitekturer med "softwarebusser" som OMG CORBA og Microsoft COM vundet udbredelse. Derigennem består applikationer i stigende grad af uafhængige, eksekverbare komponenter, der hver især uden videre kan videreudvikles og udskiftes forudsat at grænsefladerne og den grundlæggende funktionalitet bevares uændret. Disse softwarekomponenter er dog primært passive programmeringskomponenter som programmører binder sammen i programmer der kontrollerer softwarekomponenterne.

Målet for Object Management Group er at videreudvikler CORBA softwarekomponenter fra at være programmerbare til også at være konfigurerbare, således at slutbrugere/superbrugere selv kan udvikle applikationer bestående af forbundne softwarekomponenter. Specifikationerne for dette er stadig under udarbejdelse men det foreløbige forslag omfatter anvendelse af flere forskellige alternativer i form af CORBAscript og en række andre, eksisterende scripting programmeringssprog som Javascript, Tcl og Python. Konfigurering af applikationer bestående af et antal eksisterende komponenter kræver således stadig en vis programmeringsforståelse men kan eventuelt også ske indirekte med visuelle værktøjer tilgængelige for almindelige slutbrugere.

Linda er et generelt koordineringsprogrammeringssprog som supplement til andre programmeringssprog i form af yderligere funktioner til koordinering og kommunikation mellem forskellige programmer, der eventuelt er programmeret i forskellige programmeringssprog. Koordineringssproget er formuleret med udgangspunkt i en generel koordineringsmodel, der omfatter alle nødvendige koordineringsmekanismer i såvel simple som parallelle og distribuerede maskinarkitekturer. Som generel koordineringsmodel omfatter/erstatter Linda traditionelle koordinerings- og kommunikationsmekanismer som Remote Procedure Calls samt traditionel styresystem funktionalitet som filsystemer og datakommunikation over netværk.

Flow-baseret programmering omfatter et simpelt datafloworienteret koordineringsprogrammeringssprog, der giver slutbrugere mulighed for selv at konfigurere applikationer bestående af softwarekomponenter forbundet igennem et netværk af inddata og uddata. Flow-baseret programmering består for slutbrugerne/superbrugerne udelukkende af konfigurering af forbindelser mellem softwarekomponenternes inddata og uddata og kræver dermed umiddelbart ingen programmeringsforståelse for kontrolstrukturer med mere - dette ligger udelukkende i softwarekomponenterne.

Programmer's Playground er ligeledes et datafloworienteret koodineringsprogrammeringssprog, og i forbindelse med dette er der desuden udviklet brugergrænseflade og applikation styringssystemer, som giver slutbrugere mulighed for selv at konstruere/konfigurere avancerede, distribuerede, multimedia applikationer med komplekse datatyper/-strukturer, brugergrænseflader og maskinarkitekturer.

Konklusion på interessante idéer og udviklinger indenfor systemmodellering

prev up level

Integreret systemmodellering

Formalisering

Flow orientering

Udbredelsen af objektorienterede koncepter til alt fra systemanalyse til distribuerede maskinarkitekturer viser at det i vid udstrækning er muligt at anvende fælles systemmodellerings koncepter igennem hele systemudviklingsforløbet for bedre integrerede og mere effektive systemudviklingsforløb. Standardiseringen af objektorienterede koncepter og tilhørende systemmodeller for systemudviklingsværktøjer og maskinarkitekturer viser desuden udbyttet ved i et vist omfang frit at kunne kombinere/integrere iøvrigt uafhængige produkter.

Der er dog stadig lang vej til én forenet objektorienteret systemmodellering idet det i en vis udstrækning fortsat er nødvendigt med forskellige systemmodeller for henholdsvis analyse, design og implementering samt forskellige systemmodelleringssprog for henholdsvis analyse/design og implementering på trods af de fælles systemmodellerings koncepter.

Integrationen af formelle metoder med traditionel systemudvikling viser at præcision og brugervenlighed/tilgængelighed ikke nødvendigvis er modsætninger, men at selv meget forskellige systemmodelleringsmetoder kan integreres igennem formulering af en fælles, grundlæggende semantik. Derved er der adgang til et bredere udvalg af systemudviklingsmetoder i forbindelse med de enkelte systemudviklingsaktiviteter samt mulighed for bedre integreret systemmodellering på tværs af de forskellige systemudviklingsaktiviteter.

Højniveau formalisering i form af problem, analyse og design mønstre viser at systemmodelleringssprog ikke nødvendigvis blot er begrænset til nogle få, simple systemmodelleringselementer som for eksempel data, processer og objekter. Med relativt simple systemmodelleringssprog som for eksempel UML er det uden videre muligt at formulere problem, analyse og design mønstre, der fungerer som en form for systemmodelleringssprog på et højere abstraktionsniveau.

Den stigende grad af flow orientering i systemer og systemudvikling er indtil videre forholdsvis uformel og spredt, men den viser dog at flow orientering med opdeling af systemer i separate, kommunikerende systemelementer er hensigtsmæssig og eventuelt nødvendig til håndtering af komplekse systemer.

Objektorienteret systemudvikling er også i en vis grad flow orienteret i forhold til for eksempel struktureret systemudvikling med primært enkeltstående, monolitiske programmer. Objektorienterede systemer er dog kun delvist flow orienterede idet objekters eventuelt mange forskellige processer/metoder fortsat er samlet omkring det pågældende objekt, og de enkelte processer/metoder er i vid udstrækning implementeret som sekventielle programmer. Rendyrket flow orientering kan eventuelt være hensigtsmæssig eller nødvendig til udbygning af eller erstatning for objektorientering.

Generel systemmodellering

next up level

Systemvidenskab

Computerbaserede systemer

Generelt systemmodelleringssprog

Generel systemmodellering foreslåes her som fundament for systemmodellering i forbindelse med systemudvikling af computerbaserede systemer. I det følgende præsenteres centrale koncepter og teorier fra systemvidenskab og disses relation til computerbaserede systemer med efterfølgende formulering af et generelt systemmodelleringssprog til systemmodellering af computerbaserede systemer.

Systemvidenskab omfatter generelle teorier for systemer som en tværfaglig disciplin for organiserede og komplekse systemer indenfor forskellige systemparadigmer med fokus på henholdsvis "hårde", maskinelle og naturlige systemer samt bløde og kritiske, sociale systemer. Systemvidenskab omfatter desuden teorier for håndtering og simplificering af sådanne komplekse systemer, samt mere specialiserede teorier for forskellige typer af systemer. Endeligt omfatter systemvidenskab teorier for systemmodellering og systemanalyse af komplekse systemer.

Med fokus på organiserede og komplekse systemer er generel systemteori meget relevant i forbindelse med computerbaserede systemer, der har mange relationer til generel systemteori, og generel systemteori anvendes allerede og i stadig stigende grad i forbindelse med systemudvikling af computerbaserede systemer.

Med udgangspunkt i grundlæggende koncepter og teorier for generel systemteori omfattende system-relation, type-instance og whole-part systemelementer og relationer formuleres der et generelt systemmodelleringssprog specielt med henblik på systemmodellering af computerbaserede systemer, og det vises hvordan dette generelle systemmodelleringssprog kan anvendes til systemmodellering og systemimplementering af data og processer i computerbaserede systemer.

Systemvidenskab

next up down

Systembegrebet

Generel systemteoris genstandsområde

Systemparadigmer

Systemkompleksitet og simplificering

Generelle systemtyper

Generelle systemmodelleringssprog

Generelle systemteorier

Systemvidenskab er et forholdsvist nyt og meget bredt og fragmenteret fagområde, og i det følgende udvælges og præsenteres centrale, grundlæggende dele af fagområdet med relevans for generel systemteoris anvendelighed i forbindelse med systemudvikling af computerbaserede systemer.

Systembegrebet anvendes i til dagligt og indenfor nærmest alle fagområder om utallige forskellige slags ting som regel uden nærmere definitioner. Indenfor systemvidenskab diskuteres og defineres systembegrebet derimod nærmere som udgangspunkt for forskellige generelle og specialiserede systemteorier.

Med et meget bredt systembegreb omfatter systemvidenskab i et vist omfang alle andre fagområder men dermed er målet med systemvidenskab ikke altomfattende teorier. Generel systemteori fokuserer derimod udelukkende på det generelle og tværgående og overlader de specialiserede ting til de enkelte fagområder.

Et fagområde som systemvidenskab med generelle systemteorier kunne man eventuelt forvente var karakteriseret ved et fælles teoretisk grundlag, men dette er ikke tilfældet. Systemvidenskab er i hvert fald indtil videre i vid udstrækning selv specialiseret/fragmenteret i et antal forskellige fagområder/paradigmer uden tværgående integration.

Den primære egenskab ved systemer indenfor systemvidenskab er organiseret kompleksitet og centrale, grundlæggende, generelle systemteorier omfatter håndtering og simplificering af sådan kompleksitet i systemer og systemmodeller.

Generel systemteori omhandler ikke blot en enkelt, generel systemtype men også mange forskellige mere specialiserede systemtyper. Disse er dog stadig tværgående, generelle systemtyper der omfatter mange forskellige slags konkrete systemer med forskellige, generelle, systemiske egenskaber.

Ligesom systemvidenskab ikke omfatter et enkelt systembegreb omfatter det heller ikke ét generelt systemmodelleringssprog men derimod en række forskellige typer og implementeringer fra ideelle, matematiske/logiske modelleringssprog til pragmatiske, uformelle verbale sprog.

Systemvidenskab omfatter mange forskellige generelle systemteorier for systemer og systemiske egenskaber, systemmodeller og systemmodellering, samt systemudviklingsprocesser.

Systembegrebet

next up

Bertalanffy [1968] General System Theory

Weinberg [1975] Introduction to General Systems Thinking, An

Klir [1991] Facets of Systems Science

Zeigler [1976] Theory of Modelling and Simulation

Systembegrebet anvendes meget bredt i forbindelse med mange forskellige ting fra naturlige fænomener over menneskeskabte fænomener til fiktive/virtuelle fænomener, og systembegrebet kan ikke defineres i forhold til noget specifikt fænomen. Systembegrebet er konstruktivistisk og kan anvendes om alt der kan betragtes systemisk udfra beskuernes synsvinkel.

Et system kan defineres som et sæt af "ting" og relationer mellem disse, hvor tingene og typen af relationer kan være hvad som helst. Det eneste krav til et system er at tingene på en eller anden måde er bundet sammen af en form for relationer.

I forbindelse med systemer kan der desuden skelnes mellem tingene og relationer der er henholdsvis indenfor og udenfor systemet, hvor tingene udenfor systemet kan betragtes som systemets omgivelser. Systems omgivelser kan eventuelt begrænses til kun at omfatte systemets umiddelbare omgivelser i form af ting med direkte/tætte relationer til systemet.

De enkelte ting i et system kan selv betragtes som systemer, således at systemer kan ses som bestående af principielt uendelige, hierarkiske delsystemer.

Idet systembegrebet er konstruktivistisk og kun afspejler én synsvinkel på de omfattede ting og relationer, kan der desuden defineres andre relevante systemer der helt eller delvist omfatter de samme ting og relationer. Systemer kan dermed også ses som flerdimensionelle, hierarkiske delsystemer, hvor de enkelte ting og relationer kan indgå i flere forskellige delsystemer.

En mere snæver/konkret definition af (dynamiske) systemer er et matematisk koncept bestående af en tidsakse; et antal matematiske sæt der karakteriserer alle mulige indgående stimuli og udgående resultater; samt to matematiske funktioner for henholdsvis transformation af systemets interne tilstand og generering af systemets udgående resultater.

Der findes ikke ét universielt systembegreb, men mange forskellige konkrete formuleringer og implementeringer afhængigt af anvendelsesområdet. Der anvendes dog en række gennemgående, grundlæggende koncepter med hierarkier af (del)systemer og relationer mellem disse. I forbindelse med dynamiske systemer er de typisk præciseret til en form for input-output relationer og transformerende (del)systemer eventuelt suppleret med interne tilstande i de enkelte (del)systemer.

Generel systemteoris genstandsområde

prev next up

Bertalanffy [1968] General System Theory

Klir [1991] Facets of Systems Science

Zeigler [1976] Theory of Modelling and Simulation

Weinberg [1975] Introduction to General Systems Thinking, An

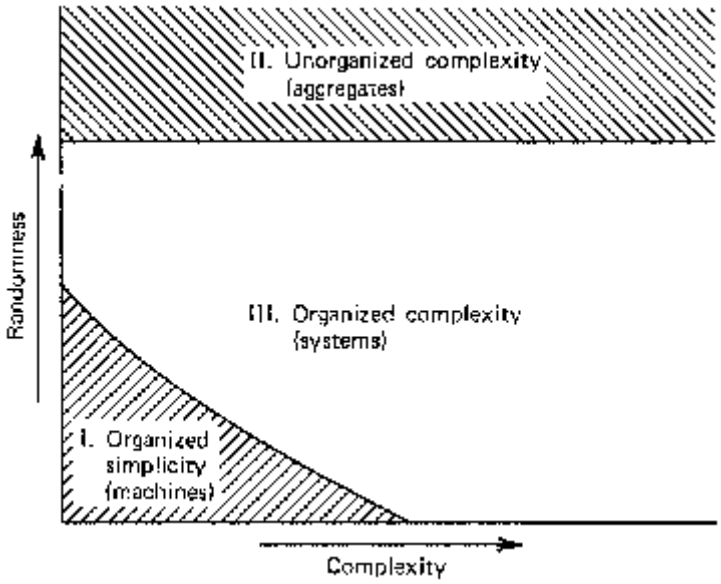
Såvel generel systemteori som traditionelle fagdiscipliner beskæftiger sig i vid udstrækning med systemer. Generel systemteori tager dog udgangspunkt i og fokuserer på relationerne i et system mere eller mindre uafhængigt af hvad "tingene" i et system er, hvorimod traditionelle fagdiscipliner tager udgangspunkt i og fokuserer på tingene i et system i form af fysiske fænomener. Generel systemteori er således en tværfaglig disciplin på linie med matematik og statistik.

I forhold til matematik og statistik er generel systemteori placeret et sted imellem dem. Matematik omhandler organiseret simplicitet i form af lovmæssige relationer for forholdsvist få variable data. Statistik omhandler uorganiseret kompleksitet i form af sandsynligheds relationer for mange variable data. Generel systemteori omhandler organiseret kompleksitet i form af lovmæssige relationer for forholdsvist mange variable data.

Organiseret simplicitet omfatter systemer med et mindre antal elementer med simple, rent deterministiske indbyrdes relationer således at det er muligt at beskrive og bearbejde dem analytisk med matematiske metoder. Med et større antal elementer med komplekse deterministiske indbyrdes relationer og eventuelt ikke-deterministiske indbyrdes relationer er det praktisk umuligt at beskrive og bearbejde dem analytisk med matematiske metoder.

Uorganiseret kompleksitet omfatter derimod systemer med et vilkårligt antal elementer uden eller med ikke-deterministiske indbyrdes relationer således at det er muligt at beskrive og bearbejde dem som en masse med statistiske metoder. Med helt eller delvist deterministiske indbyrdes relationer mellem elementerne er statistiske metoder dog ikke eller kun begrænset anvendelige.

Organiseret kompleksitet omfatter således de resterende systemer med et større antal elementer og helt eller delvist deterministiske indbyrdes relationer som ikke umiddelbart kan beskrives og bearbejdes matematisk eller statistisk i tilstrækkelig grad.



Kilde: Weinberg (1975). An Introduction to General Systems Thinking.

Generel systemteori omhandler desuden i vid udstrækning systemiske egenskaber, der ikke kan henføres til de enkelte delsystemer, men som kun opstår i systemer som helhed som et resultat der fremkommer af relationerne mellem tingene i systemet.

Systemparadigmer

prev next up

Bertalanffy [1968] General System Theory

Klir [1991] Facets of Systems Science

Checkland [1990] Soft Systems Methodology in Action

Flood & Jackson [1992] Critical Systems Thinking

Generel systemteoris udganspunkt var oprindeligt matematisk orienteret modellering/formulering af komplekse systemer, men dette har senere udviklet sig fra "hårde", matematisk orienterede systemer til "bløde", sociale systemer og "kritiske", politiske systemer.

Målet med "hård" systemteori er i sidste ende at formulere eksakte, matematiske lovmæssigheder for komplekse systemer, som objektive fænomener. Hård systemteori omfatter blandt andet teorier om stabilitet, variation, kontrol, regulering og selvregulering i systemer, samt modellering og analyse af systemer igennem sæt-, graf- og netteori.

"Blød" systemteori orienterer sig derimod om anvendelige men ikke nødvendigvis sande modeller og teorier for sociale systemer, som subjektive fænomener. Blød systemteori omfatter teorier for modellering og udvikling af sociale systemer (human activity systems) igennem opbygning af fælles forståelse, målfastsættelse og handling.

"Kritisk" systemteori fokuserer på en nødvendig, kritisk indgangsvinkel til modellering og forandring af systemer, som politiske fænomener og processer. Kritisk systemteori omfatter teorier for kritisk vurdering af antagelser, værdier, systemteorier, -metoder og -metodologier samt metametodologisk valg og anvendelse af systemmetodologier - herunder såvel hårde som bløde systemmetodologier.

Generel systemteori dækker dermed et meget bred område og meget forskellige indgangsvinkler til systemer, og de forskellige systemparadigmer kan/bør i vid udstrækning måske snarere ses som supplerende delfagområder, der fokuserer på forskellige typer af og problemer ved systemer med hård systemteori orienteret mod modellering og simulering af naturlige og mekaniske systemer, og blød/kritisk systemteori orienteret mod sociale og politiske systemer samt systemudviklingsprocesser. Én generel systemteori burde principielt dække hele spektret, men indtil videre er der ikke formuleret ét integreret systemparadigme, der detaljeret omfatter det hele.

Systemkompleksitet og simplificering

prev next up

Weinberg [1975] Introduction to General Systems Thinking, An

Klir [1991] Facets of Systems Science

Zeigler [1976] Theory of Modelling and Simulation

Et helt centralt koncept i forbindelse med generel systemteori er kompleksitet, idet generel systemteoris genstandsområde netop er organiseret kompleksitet. Generel systemteori omfatter således teorier for kompleksitet og simplificering for håndtering af kompleksitet i systemer.

Et systems kompleksitet kan generelt karakteriseres ved systemets deskriptive kompleksitet i form af antallet og variationen af enhederne og relationerne systemet består af; samt systemets usikkerhed i forhold til systemets forudsigelser/udtryksfuldhed. Matematisk baserede systemer kan desuden karakteriseres ved systemets beregningskompleksitet i form af den tid og resourcer det tager at "løse" systemet.

I forbindelse med simplificering af systemer angiver generel systemteori en uundgåelig afvejning mellem deskriptiv kompleksitet og usikkerhed-baseret kompleksitet, idet reduktion af den ene øger den anden. Simplificering må således foretages som afvejning mellem den ene eller anden form for kompleksitet.

Simplificering af systemer omfatter generelt ekskludering af variable, gruppering/klassificering af variable og opdeling i delsystemer.

Ekskludering af elementer/relationer fra systemet begrænser naturligvis systemets beskrivelsesevne og kan således kun anvendes i begrænset omfang uden at systemet mister sin anvendelighed, men ekskludering af unødvendige elementer/relationer kan simplificere systemer væsentligt med hensyn til deskriptiv kompleksitet og beregningskompleksitet.

Begræsning af systemers deskriptive kompleksitet og beregningskompleksitet kan desuden begrænses ved at klassificere/kvantificere variabel værdier og behandle dem som ét. Dette begrænser dog systemets præcision og øger dermed usikkerheden.

Endelig er opdeling af et system i delsystemer en måde at mindske den deskriptive kompleksitet og beregningskompleksiteten. I det omfang at relationerne mellem delsystemerne eventuelt begrænses ved opdelingen i delsystemer medfører det begrænset udtryksfuldhed og øget usikkerhed. Opdeling i delsystemer uden begrænsede relationer mellem de enkelte delsystemer begrænser ikke den totale kompleksitet i systemet, men opdelingen i mindre komplekse delsystemer kan medføre begrænset deskriptiv kompleksitet og beregningskompleksitet samt væsentligt simplere håndtering af kompleksisteten.

Generelle systemtyper

prev next up

Boulding [1956] General Systems Theory - The Skeleton of Science

Klir [1991] Facets of Systems Science

Zeigler [1976] Theory of Modelling and Simulation

Med generel systemteoris brede genstandsområde med principielt alle systemer angiver generel systemteori modeller for kategorisering af systemtyper efter systemers indhold i form af relationer.

Systemer kan kategoriseres i et hierarki af systemtyper udfra systemernes kompleksitet fra forholdsvist simple statiske struktursystemer til sociale systemer (og transcendentale systemer). Statiske struktursystemer omfatter uforanderlige systemer med statiske relationer mellem elementerne. Simple dynamiske systemer omfatter systemer med deterministiske relationer mellem elementerne. Kontrolsystemer omfatter systemer der regulerer sig selv udfra fra givne mål. Selvorganiserende systemer omfatter systemer der vedligeholder sig selv igennem optagelse og udskillelse af materiale til/fra systemets omgivelser. Genetisk-sociale systemer omfatter systemer bestående af specialiserede delsystemer og indbyggede gener, der bestemmer systemets udvikling. Selvorienterende (self-awareness) systemer omfatter systemer der orienterer sig og reagerer i forhold til systemets omgivelser. Selvbevidste systemer omfatter systemer med selvrefleksion og symbolsk fortolkning/forståelse. Sociale systemer omfatter systemer bestående af selvstændige delsystemer der påvirker hinanden igennem symbolsk kommunikation og handling.

En anden dimension for kategorisering af systemer er i et såkaldt epistemologisk hierarki af systemer. Kildesystemer definerer systemers datatyper og dataværdier og afgrænser derved de "ting" systemet omfatter i form af for eksempel input og output variable. Datasystemer består af empiriske data aflæst for konkrete forekomster af systemer. Genererede systemer (generative system) består af genererede data for mulige/forventede konkrete forekomster af systemer i form af for eksempel simulering og estimering. Struktursystemer består af relationer i systemer i form af for eksempel beregning af output variable udfra input variable. Metasystemer består af styring i systemer for valg, skift eller transformation mellem systemer i form af for eksempel valg af relevant struktursystem udfra input variable.

Et tilsvarende hierarki kategoriserer ligeledes systemer fra enkeltstående input-output observationer til netværk af input-output transformerende systemer som henholdsvis input-output relation observationer; input-output funtion observationer; input-output system abstraktioner; iterative specifikationer; strukturerede system specifikationer; og netværk af input-output systemer.

De matematisk orienterede formelle systemhierarkier definerer primært, basale systemer som forskellige former for input-output transformerende processer samt netværk af disse til mere komplekse systemer. Disse sammensatte systemer kan videre kategoriseres ud fra deres funktion som forskellige former for selvregulerende systemer, der udspringer af interaktionen mellem de enkelte delsystemer.

Generel systemteori omfatter således meget forskellige systemtyper med hver sine relevante, mere specialiserede systemteorier, men disse forskellige systemtyper bygger samtidigt alle på simple, fælles, grundlæggende systemkoncepter og basale systemtyper.

Generelle systemmodelleringssprog

prev next up

Bertalanffy [1968] General System Theory

Weinberg [1975] Introduction to General Systems Thinking, An

Jackson [1995] Software Requirements & Specifications

Generel systemteoris oprindelige mål er i sidste ende matematisk modellering af komplekse systemer på linie med naturvidenskabernes matematiske modellering af simple systemer for eksakt modellering af systemer. Dette er dog ikke en forudsætning for generel systemteori, og uformel modellering ses som nyttig og eventuel nødvendig for modellering af komplekse systemer før en eventuel mere formel modellering.

Modellering med matematisk algebra giver mange fordele i form af præcision og metoder til analyse og bearbejdning af modellerne men stiller omvendt også store krav til præcision; resulterer i meget abstrakte modeller og er ikke velegnede til alle typer af systemer.

Et alternativt/komplementerende formelt modelleringssprog er prædikativ logik, der ligeledes muliggør stor præcision, og som umiddelbart er lettere tilgængelig og velegnet til nogle typer af systemer.

Matematisk sæt og net teori er ligeledes i vid udstrækning anvendelig til modellering af systemer og muliggør stor præcision og matematisk analyse og bearbejdning.

Ad-hoc diagrammatisk modellering uden eksakt semantisk formalisering er fortrinsvis uformel. Der er dog en vis form for formalisering igennem konsistent anvendelse af et antal diagrammatiske elementer indenfor en model.

Uformelle verbale beskrivelser er desuden nødvendige i forbindelse med designationer for formelle systemer og beskrivelse af aspekter der ikke umiddelbart er formaliserbare. Uformelle verbale beskrivelser er ligeledes hensigtsmæssige for umiddelbart lettere forståelige beskrivelser.

Generelle systemteorier

prev up

Bertalanffy [1968] General System Theory

Weinberg [1975] Introduction to General Systems Thinking, An

Weinberg [1988] Rethinking Systems Analysis & Design

Weinberg [1988] General Principles of Systems Design

Zeigler [1976] Theory of Modelling and Simulation

Checkland [1990] Soft Systems Methodology in Action

Flood & Jackson [1992] Critical Systems Thinking

Generel systemteori og relaterede matematiske discipliner tilbyder en lang række generelle teorier og metoder for systemer og håndtering af systemer.

Generel systemteori omfatter teorier for systemiske egenskaber i form af vurdering og simplificering af kompleksitet og usikkerhed, systemtyper, opstående (emergent) egenskaber samt struktur og regulering i systemer.

Generel systemteori omfatter ligeledes processorienterede teorier for håndtering af systemer i form af observation, analyse og design af systemer, der angiver generelle muligheder og begrænsninger for observation og analyse af systemer, samt generelle principper for generalisering og design af systemer med håndtering af usikkerhed og kompleksitet.

Systemer kan i vid udstrækning modelleres som matematiske sæt og net, hvilket muliggør matematisk analyse og bearbejdning af modellerne. Dette omfatter identificering og beregning af kvantificerbare systemisk egenskaber og transformation mellem forskellige systemmodel præsentationer. Matematisk baserede systemmodeller muliggør desuden simulering med henblik på validering af systemmodeller og eksperimenter med forskellige systemmodeller.

Blød og kritisk systemteori tilbyder ligeledes en vis formalisering af sociale systemer i form af teorier for modellering, analyse og udvikling af sociale systemer.

Generel systemteori omfatter således en bred vifte af meget forskellige systemteorier med formelle matematiske systemteorier orienteret mod modellering og validering af selve systemmodellerne, samt mere uformelle systemteorier orienteret mod systemmodellering og systemudvikling processer og validering af systemmodeller i forhold til system omgivelserne.

Computerbaserede systemer

prev next up down

Computerbaserede systemer vs. generel systemteori

Computerorienterede systemtyper

Computerorienteret systemmodellering

Computerorienterede systemteorier

Generel systemteori er et tværfagligt fagområde med fokus på systemer generelt uafhængigt af typen af fænomener systemet omfatter, men forskning og anvendelse sker dog i vid udstrækning i forbindelse med andre fagområder. I det følgende beskrives udvalgte relationer mellem generel systemteori og systemudvikling af computerbaserede systemer.

Ikke-trivielle computerbaserede systemer udgør nærmest arketypen af komplekse systemer med næsten utallige, gensidigt afhængige variable og komplekse relationer, og de absolut mest komplekse menneskeskabte systemer består af computerbaserede systemer. Generel systemteori bør således være meget relevant for systemudvikling af computerbaserede systemer.

Traditionel systemudvikling har primært fokuseret på systemudvikling metoder og værktøjer med udgangspunkt i computeren uden videre skelnen mellem forskellige systemtyper med hensyn til systemernes indhold og funktion. Dette er dog under hastig forandring med udbredelsen af anvendelse af mønstre, hvorved systemudvikling af computerbaserede systemer på tilsvarende måde som generel systemteori karakteriserer og kategoriserer forskellige (del)systemer udfra deres generelle, systemiske egenskaber.

Med computere som et centralt emne samt værktøj for generel systemteori er det naturligt at der formuleres generelle systemteorier for systemmodellering af dynamiske processer og generelle maskiner som computere. Disse systemteorier er dog primært fokuseret på anvendelse af computere som værktøj for generel systemteori med henblik på systemmodellering og simulering af systemer og generelle systemteorier.

Generel systemteori omfatter dog også en række teorier og metoder til systemmodellering og systemanalyse, der umiddelbart er anvendelige og i vid udstrækning også allerede anvendes i forbindelse med systemudvikling af computerbaserede systemer.

Computerbaserede systemer vs. generel systemteori

next up

Generel systemteoris genstandsområde

Systemparadigmer

Integreret systemmodellering

Formalisering

Flow orientering

Klir [1991] Facets of Systems Science

Stowell [1995] Does 'Information Systems' Need Systems?

Ligesom generel systemteoris genstandsområde kan computere og computerbaserede systemer i høj grad karakteriseres som organiseret kompleksitet i form af generelle, formaliserede maskiner med mange variable.

Helt grundlæggende fungerer computere og computerbaserede systemer som simple maskiner der uden videre kan beskrives matematisk/logisk for det enkelte operationer. Ved udvikling af ikke-trivielle computerbaserede programmer stiger kompleksiteten dog hurtigt med mange gensidigt afhængige variable og deraf fremkommende systemiske egenskaber. Store computerbaserede programmer er således også i vid udstrækning karakteriseret ved at indeholde mange fejl og ændringer af dele af et program kan have væsentlige, uigennemskuelige konsekvenser for andre dele af programmet.

Computeren er også et centralt værktøj for generel systemteori som generel maskine for modellering og simulering/eksekvering af vilkårlige systemer samt som eksperimentielt laboratorium for analyse og design af generelle systemer.

Den centrale rolle for computeren som værktøj for generel systemteorier skyldes dels computeres meget store beregningskapacitet, der muliggør matematisk bearbejdning af mange variable med mange og komplekse indbyrdes relationer. Derudover er computeren en universel maskine der kan programmeres til at fungere som en virtuel maskine for en hvilken som helst anden formel/formaliseret maskine.

Systemudvikling af computerbaserede systemer karakteriseres ofte som et multidisciplinært fagområde, men der er stillet spørgsmålstegn ved om informatiks multidisciplinære udgangspunkt i teorier fra forskellige meget forskellige fagområder afspejler et manglende teoretisk fundament for informatik, og generel systemteori er foreslået som dette forenende fundament.

Systemudvikling af computerbaserede systemer omfatter ligesom systemvidenskab meget forskellige aspekter fra eksakt modellering af konkrete, eksekverbare computer programmer med "hård" generel systemteori relationer til socialt orienteret systemudvikling med "blød" og "kritisk" systemteori relationer.

Indenfor informatik og computerbaseret systemudvikling er der desuden forskellige udviklingstendenser i retning af øget generalisering med bevægelse i retning af fælles modelleringskoncepter for forskellige abstraktionsniveauer og systemudviklingsaktiviteter; integration af forskellige formelle og uformelle modelleringssprog; samt flow-orientering for både maskinarkitektur, programmering og applikationer.

Der er således mange og tætte relationer mellem generel systemteori og systemudvikling af computerbaserede systemer, og i stedet for at systemudvikling af computerbaserede systemer eventuelt langsomt bevæger sig i retning af generel systemteori, er det måske muligt og hensigtsmæssigt helt at reformulere systemudvikling af computerbaserede systemer med udgangspunkt i generel systemteori.

Computerorienterede systemtyper

prev next up down

Mønstre

Generelle systemtyper

Bouldings generelle systemtyper

Boulding [1956] General Systems Theory - The Skeleton of Science

Jackson [1995] Software Requirements & Specifications

Systemudvikling har længe foregået uden videre formalisering end simple grundlæggende koncepter som trinvis, funktionel nedbrydning og separation og komplementaritet mellem data, processer og tilstande. På højere abstraktionsniveauer har systemudvikling i vid udstrækning været overladt til "naturlig" strukturering. Med udbredelsen af mønstre i forbindelse med systemudvikling har dette dog ændret sig væsentligt med en form for formalisering af forskellige typer af (del)systemer.

Der er mange ligheder mellem problemrammer og generelle systemtyper, og en del af de enkelte problemrammer omhandler de samme problemstillinger som tilsvarende generelle systemtyper. Dette gælder specielt for de simple, mekanistiske, generelle systemer men også i et vist omfang for de komplekse, organistiske, generelle systemer, der har ligheder med komplekse computerbaserede systemtyper.

Generelle simple dynamiske systemer omfatter systemer med deterministiske relationer mellem elementerne, hvilket omfatter simpel input-output transformation. JSP (Jackson Structured Programming) problemrammer omfatter tilsvarende modellering af et system til transformation af strukturerede inddata til uddata.

Generelle kontrolsystemer omfatter systemer der regulerer sig selv udfra fra givne mål i form af feedback og feedforward systemer. Simple kontrol problemrammer omfatter tilsvarende systemer.

Generelle genetisk-sociale systemer omfatter systemer bestående af specialiserede delsystemer og indbyggede gener, der bestemmer systemets udvikling. Værktøj (workpieces) problemrammen omfatter systemer der reagerer på eksterne stimuli og udfra disse udvikler sig selv eller en selvstændig model udfra indbyggede regler.

Generelle selvorienterende (self-awareness) systemer omfatter systemer der orienterer sig og reagerer i forhold til systemets omgivelser. Simple system problemrammer omfatter observationer i form af inddata, der fortolkes i forhold til en model af det observerede fænomen for at levere kvalificerede uddata samt til opbygning og vedligeholdelse af modellen.

Ved at kombinere generelle systemteorier for de generelle systemtyper og yderligt specialiserede typer af disse med problemrammer kan der opnåes en bedre forståelse af de forskellige problemstillinger og karakteristika ved systemtyperne. Yderligere detaljering af systemtyperne vil desuden kunne føre naturligt og struktureret til relevante analyse- og designmønstre samt algoritmer, således at det resulterer i et bedre integrerede systemudviklingsforløb for udvikling af computerbaserede systemer.

Computerorienteret systemmodellering

prev next up

Zeigler [1976] Theory of Modelling and Simulation

Zeigler & Vahie [1993] DEVS Formalism and Methodology

Jensen [1997] A Brief Introduction to Coloured Petri Nets

Jensen [1994] An Introduction to the Theoretical Aspects of Coloured Petri Nets

Jensen [1996] An Introduction to the Practical Use of Coloured Petri Nets

Idet en computer kan betragtes som en universel maskine der kan programmeres til at fungere som en hvilken som helst anden formel/formaliseret maskine, bør det også være muligt at formulere en generel beskrivelse af computere og deres virkemåde.

Et eksempel på en sådan generel formulering er en Turing maskine, som netop dannede udgangspunkt for udviklingen af computere. En Turing maskine er dog ikke praktisk anvendelig til komplekse systemer. Andre generelle systemteorier for systemmodellering af dynamiske systemer som DEVS og Petri Net er orienterede mod simulering og er dermed heller ikke umiddelbart egnede til implementering af direkte eksekverbar computer programkode. Det bør dog tilsvarende være muligt at beskrive computere og deres virkemåde på andre generelle måder orienteret mod implementering af direkte eksekverbar computer programkode.

Tilsvarende er lavniveau maskinekodeprogrammering af computere ikke praktisk anvendelig til komplekse systemer, hvorfor der i vid udstrækning anvendes højniveau programmeringssprog og andre højniveau systemmodelleringssprog til systemudvikling af computerbaserede systemer. Det centrale for sådanne højniveau systemmodelleringssprog er anvendelsen af abstraktioner for ofte benyttede konstruktioner og skjulning af unødvendige detaljer. Indenfor generel systemteori er opdeling og opbygning af systemer i/med delsystemer ligeledes centralt for håndtering og eventuelt reducering af systemkompleksitet.

Fremfor at benytte mange forskellige specielle systemmodelleringssprog med hver sine abstraktioner, som det i vid udstrækning i dag er tilfældet i forbindelse med systemudvikling af computerbaserede systemer, bør der kunne formuleres generelle abstraktionsmekanismer, der giver tilsvarende muligheder for at formulere højniveau abstraktioner.

De mange forskellige specielle systemmodelleringssprog omfatter som regel abstraktionsmekanismer til hierarkisk opdeling af systemmodeller i systemer og delsystemer. Derudover omfatter mange specielle systemmodelleringssprog ligeledes abstraktionsmekanismer i form af klassificering og systemmodellering af abstrakte systemtyper og konkrete forekomster af (del)systemer. Disse to former for generelle abstraktionsmekanismer går således igen i mange specielle systemmodelleringssprog samt indenfor generel systemteori.

Generelle systemteorier for systemmodellering af dynamiske systemer som DEVS og Petri Net er matematisk baserede af hensyn til at opnå den nødvendige detaljering og præcision for at kunne anvende systemmodellerne til simulering. Dette er tilsvarende nødvendigt med henblik på at implementere direkte eksekverbar computer programkode. Det er dog typisk kun en del af en systemmodel for et computerbaseret system der er direkte computerbaseret mens at andre dele af systemmodellen omfatter domænet og omgivelserne som for eksempel i form af et social system. Til systemmodellering af dette er matematisk detaljering og præcision eventuelt ikke muligt og hensigtsmæssigt. Som for generelle systemteorier selv er matematisk præcision ønskelig men ikke altid mulig og hensigtsmæssig, og eksisterende generelle systemmodelleringssprog som DEVS og Petri Net er således eventuelt ikke velegnede. Det bør dog være muligt at formulere et generelt systemmodelleringssprog der muliggør matematisk præcision uden dog at kræve det således at det også muliggør uformel, tvetydig og inkonsistent systemmodellering for i hvert fald dele af systemet.

Systemmodellering af computerbaserede systemer omfatter systemmodellering af såvel processer som data, hvor data omfatter såvel simple data som komplekse data i form af for eksempel sammensatte datastrukturer, hierarkiske og relationelle data, samt objekter. Generelle systemmodelleringssprog som DEVS og Petri Net er derimod primært orienteret mod systemmodellering af dynamiske processer og omfatter kun simple data og simple, sammensatte datastrukturer i form af strukturerede matematiske sæt. Der er dog formuleret udvidelser/varianter af Petri Net, der også omfatter komplekse hierarkiske og

relationelle datastrukturer.

Generelle systemmodelleringssprog som DEVS og Petri Net omfatter systemmodellering af dynamiske systemer som hierarkiske netværk af input-output transformerende processer. Systemmodellering af computerbaserede systemer omfatter tilsvarende systemmodeller i form af for eksempel dataflow diagrammer men også mange andre former for systemmodeller som for eksempel strukturdiagrammer, entitet-relation modeller, netværk af kommunikerende objekter, samt sekvenser af programkode instruktioner. Der anvendes således meget forskellige systemmodeller, men de kan dog alle modelleres som en form for matematiske grafer omend meget forskellige matematiske grafer. Det bør dog eventuelt være muligt at integrere dem igennem fælles graf baserede koncepter.

Der er således mange lighedspunkter og fælles koncepter mellem generel systemmodellering og systemmodellering af computerbaserede systemer men også mange forskelle specielt med hensyn til implementeringen af koncepterne. De generelle systemmodelleringssprog som DEVS og Petri Net er dermed heller ikke umiddelbart anvendelige og hensigtsmæssige til fuld systemmodellering af computerbaserede systemer. Det bør dog være muligt at formulerer et generelt systemmodelleringssprog, der er anvendeligt til fuld systemmodellering af computerbaserede systemer omfattende uformelle, sociale delsystemer såvel som formelle, maskinelle delsystemer.

Computerorienterede systemteorier

prev up

Wasserman & Faust [1994] Social Network Analysis: Methods and Applications

McHugh [1990] Algorithmic Graph Theory

Fjällström [1998] Algorithms for Graph Partitioning: A Survey

Rozenberg [1997] Handbook of Graph Grammars and Computing by Graph Transformation: Foundations, Vol. 1

Jackson [1983] System Development

Kummer & Schlange [1997] Strengthening the Bridge between Qualitative and Quantitative Modeling

Generel systemteori omfatter en række grundlæggende teorier for systemer samt deres kompleksitet og simplificering med mere, men derudover er der også mange mere specialiserede, generelle systemteorier og relaterede matematiske teorier og metoder med relevans i forbindelse med systemmodellering af computerbaserede systemer.

Systemmodeller for computerbaserede systemer kan i vid udstrækning repræsenteres i form af matematiske grafer, for hvilke der findes et stort antal matematiske teorier og metoder. Disse omfatter blandt andet netværksanalyse og graf transformationer.

Netværksanalyse af matematiske graf repræsentationer af systemmodeller for computerbaserede systemer kan for eksempel anvendes til kvantificering af netværkstætheden i grafer for de enkelte delsystemer med direkte relation til kvalitative mål for god opdeling af systemer i delsystemer i forbindelse med struktureret systemudvikling. God opdeling af systemer i delsystemer kræver således stærke bindinger indenfor de enkelte delsystemer og svage koblinger mellem de enkelte delsystemer, hvilket svarer til henholdsvis stor og lille netværkstæthed i graferne for de enkelte delsystemer.

Netværksanalyse af matematiske graf repræsentationer kan dermed anvendes til et kvantificeret mål for kvaliteten af visse aspekter i systemmodeller for computerbaserede. Netværksanalyse vil ligeledes kunne anvendes til automatisk generering af forslag til alternative, bedre opdelinger af systemer i delsystemer i forbindelse med systemmodelleringen; samt til opdeling af store, flade systemer i hierarkiske delsystemer for eksempel i forbindelse med reengineering af eksisterende eksekverbar computer programkode.

Ved repræsentation af systemmodeller for computerbaserede systemer i form af matematiske grafer kan der desuden laves forskellige transformationer af graferne og præsentationer af disse. Grafer kan reduceres med sammenfoldning af udvalgte kanter og punkter til enkelte kanter og punkter, således at uønskede detaljer kan skjules ved præsentation af grafer. Grafer kan ligeledes transformeres strukturelt med udgangspunkt i forskellige typer af systemelementer, som for eksempel fra dataflow til objekter eller tilstande.

I forbindelse med systemudvikling af computerbaserede systemer er der desuden formuleret teorier for overensstemmelse mellem hierarkiske strukturer i input/output og dynamiske systemer, således at programstrukturer kan udledes af strukturer i inddata og uddata.

Endeligt er matematisk graf baserede teorier og metoder foreslået til sammenkædning af uformelle, "bløde" systemmodeller og formelle, "hårde" systemmodeller igennem kvalitativ strukturel analyse af graf repræsentationer af bløde systemmodeller med eventuel efterfølgende kvantitative systemmodellering og simulering.

"Bløde" systemteorier er i vid udstrækning netop formuleret i forbindelse med systemudvikling af computerbaserede systemer, og disse er således umiddelbart anvendelige i forbindelse med de dele af systemudviklingen, der er orienteret mod problemidentifikation og -formulering samt systemmodellering af teknisk orienterede delsystemers omgivelser og disses relationer til tekniske orienterede delsystemer.

Generelle systemteorier er således i vid udstrækning relevante for og allerede anvendte til systemudvikling af computerbaserede systemer, og med yderligere udvikling og modning af begge fagområder vil der eventuelt fremkomme flere fælles eller relaterede systemteorier.

Generelt systemmodelleringssprog

prev up down

Generelt systemmodelleringssprog koncepter

Generelt systemmodelleringssprog udgangspunkt

System-relation koncept kerne

Type-instance koncept

Type-instance konsistens og komplethed

Whole-part koncept

Systemhierarki systemmodel

Implementering

Datamodellering

Procesmodellering

I det følgende formuleres et generelt systemmodelleringssprog specielt med udgangspunkt i at det skal kunne anvendes til systemmodellering af (delvist) computerbaserede systemer.

De nødvendige/hensigtsmæssige krav til og grundlæggende koncepter for det generelle systemmodelleringssprog identificeres som værende system-relation, type-instance og whole-part systemmodelleringskoncepter til multidimensionel systemmodellering af data og processer med mulighed for men ikke nødvendigvis krav om modellering som et formelt system.

Forskellige eksisterende systemmodelleringssprog overvejes som udgangspunkt for formuleringen af det generelle systemmodelleringssprog, hvoraf sNet findes bedst egnet. De grundlæggende systemmodelleringssprog system-relation, type-instance og whole-part koncepter defineres med en formel semantik som en på nogle områder forenklet og på andre områder udvidet version af sNet.

Endeligt beskrives hvordan det generelle systemmodelleringssprogs grundlæggende koncepter kan implementeres i forbindelse med et generelt systemmodelleringsværktøj, samt hvordan det generelle systemmodelleringssprog kan anvendes til systemmodellering af data og processer i forbindelse med computerbaserede systemer.

Generelt systemmodelleringssprog koncepter

next up

Systembegrebet

Systemkompleksitet og simplificering

Generelle systemmodelleringssprog

Generelle systemteorier

Computerorienteret systemmodellering

Computerorienterede systemteorier

Kravene til et generelt systemmodelleringssprog, der skal kunne benyttes til alle modelleringsaktiviteter i forbindelse med systemudvikling af computerbaserede systemer, er meget brede. Modelleringssproget skal være så fleksibelt at det kan anvendes til beskrivelse af såvel helt uformelle systemer som helt formelle systemer; samt konkrete, fysiske fænomener såvel som abstrakte, logiske koncepter.

Modelleringssproget må dermed nødvendigvis bestå af nogle relativt få, meget generelle koncepter, da det vil være umuligt på forhånd at formulere specielle koncepter for alle mulige systemtyper, og indtil videre er der i hvert fald ikke grundlag for at kunne begrænse sig til nogle bestemte systemtyper.

Modelleringssproget bør desuden nødvendigvis bestå af et formaliseret grundlag, der gør det muligt at validere og på anden måde at analysere modellerne udfra eventuelle yderligere kriterier i forbindelse med formelle (del)systemer.

Et generelt systemmodelleringssprog bør ikke binde systemer til bestemte koncepter medmindre at de er helt generelle. Et objektorienteret modelleringssprog er dermed formodentligt for specialiseret med blandt andet en bestemt opfattelse af relationer mellem data og processer i form af gruppering omkring data som objekter. Omvendt bør et generelt systemmodelleringssprog heller ikke lægge begrænsninger på antallet og variationen af koncepter til at sikre modellens udtryksfuldhed for at mindske usikkerheden ved modellen i forhold til det virkelige fænomen.

Det grundlæggende i et generelt systemmodelleringssprog er modellering af systemer bestående af "ting" og relationer mellem disse, hvor såvel tingene som relationer principielt kan være hvad som helst. Disse "ting" og relationer bør ligeledes kunne beskrives nærmere som systemer i sig selv eller eventuelt som aksiomatiske, numeriske eller tekstuelle data.

Et helt centralt koncept for simplificering af kompleksiteten i systemmodeller er hierarkisk abstraktion. Modelleringssproget bør derfor omfatte et "whole-part" koncept for opdeling af systemer i delsystemer. Dette er ikke begrænset til nødvendigvis at afspejle fysiske "whole-part" relationer i de modellerede fænomener men omfatter også abstrakte, koceptuelle "whole-part" relationer.

Et andet helt centralt koncept for simplificering af kompleksiteten i systemmodeller er klassificering af systemer med fælles egenskaber og fælles modellering af disse egenskaber. Modelleringssproget bør derfor omfatte et "type-instance" koncept for opdeling af systemer i systemtyper. Dette omfatter ikke også inkludering af relaterede objektorienterede koncepter som nedarvning og polyformisme. "Type-instance" konceptet er kun et modelleringsværktøj til opdeling af systemer i systemtyper og systemforekomster.

Endeligt bør et generelt modelleringssprog afspejle at systembegrebet er konstruktivistisk og derved muliggøre modellering af systemer udfra forskellige og vekslende synsvinkler. Samtidig med modellering udfra forskellige synsvinkler bør modelleringssproget understøtte integreret modellering af de forskellige synsvinklers fælles egenskaber.

I forbindelse med computerbaserede systemer og mange andre systemer er der som minimum behov for at modellere to konceptuelt forskellige systemtyper i form af henholdsvis data og processer. Den konceptuelle skelnen mellem data og processer er relevant for mange systemer men i forbindelse med computerbaserede systemer er der ikke nødvendigvis nogen grundlæggende forskel. Computerbaserede processer skal i sidste ende modelleres som programmer i form af statiske datastrukturer af instruktionskoder, der lagres og eksekveres i computeren. Modelleringssproget bør således kunne integrere de to konceptuelt forskellige data og proces synsvinkler.

Generelt systemmodelleringssprog udgangspunkt

prev next up

Bézivin [1998] Who's Afraid of Ontologies?

Lemesle [1998] Meta-modeling and Modularity

sNets [1998] sNet Formalism - Technical Report, The

Eksisterende generelle systemmodelleringssprog, som DEVS og Petri net kunne eventuelt være anvendelige til generel systemmodellering for computerbaserede systemer. Disse generelle systemmodelleringssprog er dog specielt beregnet til systemmodellering af "virkelige" systemer og simulering af disse med specielle eksekveringsmodeller tilknyttet systemmodelleringssprogene. Her er der ønske om et generelt systemmodelleringssprog der kan vendes til såvel uformelle systemmodeller som formelle, direkte eksekverbare systemmodeller i form af "almindelige" computerbaserede programmer uden brug af en simulator til eksekveringen.

Andre computer orienterede, generelle systemmodeller som Meta Object Facility (MOF) kunne eventuelt også være anvendelige til generel systemmodellering for computerbaserede systemer. MOF er dog i første omgang specielt formuleret i relation til CORBA og UML, og MOF understøtter primært koncepter der er nødvendige i forhold til disse og mangler andre systemmodellerings koncepter som for eksempel multiple typer. MOF er desuden bundet til objektorienteret systemmodellering.

Et interessant udgangspunkt for specifikation af et generelt modelleringssprog er sNet, der er foreslået som en alternativ meta-meta-model til blandt andet Meta Object Facility (MOF). sNet er baseret på semantiske netværk og består af en simpel kerne og nogle få supplerende koncepter som whole-part og type-instance samt modulariserings- og udvidelsesmekanismer til modulopdeling af systemmodeller.

Som generelt systemmodelleringssprog foreslåes dog her som udgangspunkt et i forhold til sNet lidt simplere modelleringssprog uden kardinalitetsangivelse for relationer; uden bunden, entydig navngivning af systemer for identifikation; uden bunden type angivelse for systemer; samt uden bundet modultilhørsforhold, da disse ikke umiddelbart er nødvendige og eventuelt ikke alle er hensigtsmæssige. Disse koncepter kan dog let tilføjes igen til selve modelleringssproget eller som udvidelser idet de udelukkende er defineret med modelleringssprogets kerne koncepter. Specielt vil modulopdeling konceptet være hensigtsmæssigt/nødvendigt i forbindelse med praktisk håndtering af systemmodeller. Modulopdeling kan dog eventuelt også implementeres på andre måder end i sNet.

Til gengæld er det her foreslåede generelle systemmodelleringssprog udvidet lidt i forhold til sNet med mulighed for mange-til-mange kardinalitet for relationer; og multiple typer samt multiple navne for systemer.

System-relation koncept kerne

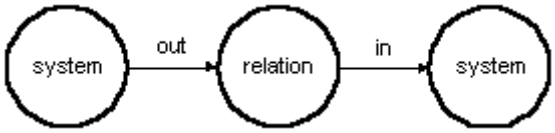
prev next up

sNets [1998] sNet Formalism - Technical Report, The

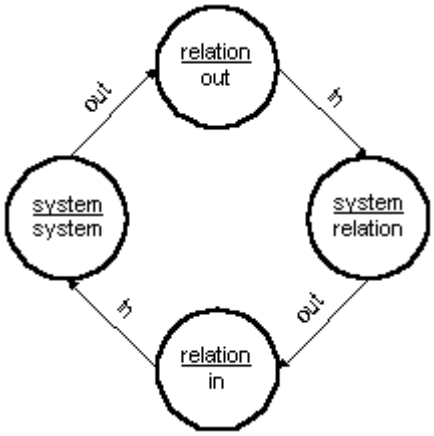
De grundlæggende elementer for det her formulerede generelle systemmodelleringssprog er meget enkle idet alle systemer udelukkende selv består af andre (del)systemer samt orienterede, boolske kanter til strukturering af (del)systemerne som matematiske grafer. Kanterne i graferne udgør ikke systemets relationer men angiver kun en eller anden form for "fører-til" / "kommer-før" ordning af (del)systemerne. Alle bestanddele af et system - såvel "ting" som relationer - betragtes i sig selv som (del)systemer. Det egentlige modelleringssprogs semantik og definitioner beskrives ved hjælp af sådanne uvægtede/ufarvede, orienterede, matematiske grafer.

Det grundlæggende koncept om systemer betående af (del)systemer og relationer udgør samtidigt hele den semantiske kerne i det generelle modelleringssprog. Kardinaliteten for relationer er som udgangspunkt "udefineret mange-til-mange" således at alle kardinaliteter er mulige.

Semantik for system-relation koncept



Refleksiv definition af system-relation koncept



Definitionsgraferne læses som navngivne noder (under stregen i noden) og kanter. Noder angives desuden med type (over stregen i noden).

Type-instance konceptet er i sig selv blot en særlig form for relationer der defineres nærmere i de følgende afsnit. System-relation og type-instance koncepterne udgør en komplet, selvdefinerende kerne for det her formulerede generelle systemmodelleringssprog. Derudover defineres i de følgende afsnit blot yderligere et whole-part koncept, der ligeledes i sig selv blot er en særlig form for relationer. Disse tre koncepter udgør den samlede, strukturelle kerne for det her definerede generelle systemmodelleringssprog.

Type-instance koncept

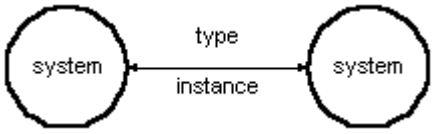
prev next up

sNets [1998] sNet Formalism - Technical Report, The

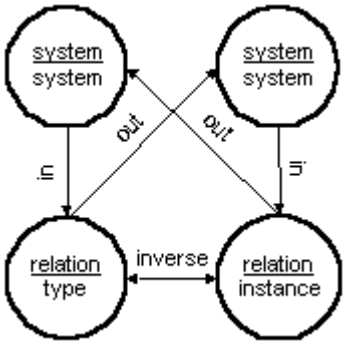
Ethvert system kan være en forekomst af et antal typer af systemer, hvis egenskaber (i form af delsystemer) systemet nedarver med hensyn til modelleringen af systemet. (Dette betyder ikke nødvendigvis nedarvning i forbindelse med implementeringen af systemet.) Ethvert system kan ligeledes være type for et antal andre system forekomster. Der er således ingen grundlæggende forskel på typer og forekomster. Type-instance relationer er gensidigt definerende og inverterede.

Type-instance relationer kan anvendes til effektiv fælles angivelse af identiske egenskaber for et antal systemer i stedet for at angive alle egenskaberne for hvert enkelt system; som skabeloner for andre systemer med angivelse af et antal faste egenskaber og strukturer; samt til kategorisering af systemer i kategorier med bestemte egenskaber.

Semantik for type-instance koncept



Definition af type-instance koncept



Nedarvning af egenskaber sker over flere niveauer fra alle overliggende typer og typers typer og så videre, og der er principielt ingen forskel på direkte nedarvning af egenskaber fra en type og indirekte nedarvning af egenskaber fra en types type.

Nedarvning af egenskaber er desuden udelukkende adderende således at alle en types egenskaber også er gældende for alle typens forekomster. Nedarvede egenskaber kan ikke ændres eller slettes for den enkelte forekomster som det for eksempel er tilfældet for objektorienterede modellerings- og programmeringssprog.

Definition af type-instance nedarvning

$\forall$ type, subtype, subsubtype

• Type-Instance(type, subtype) $\wedge$ Type-Instance(subtype, subsubtype)

$\rightarrow$ Type-Instance(type, subsubtype)

$\forall$ type, forekomst, egenskab

• Type-Instance(type, forekomst) $\wedge$ Whole-Part(type, egenskab)

$\rightarrow$ Whole-Part(forekomst, egenskab)

Type-instance konsistens og komplethed

prev next up

Ved implementering af type-instance relationer kan der skelnes mellem forskellige former for nedarvede egenskaber:

- En type kan være defineret med et antal konstante egenskaber, der er helt identiske for alle forekomster af typen.
- En type kan desuden være defineret med et antal bundne men varierende egenskaber, der forekommer i alle forekomster af typen men eventuelt med forskellige givne værdier.
- Endeligt kan en type desuden være defineret med et antal bundne men varierende egenskaber, der forekommer i alle forekomster af typen men eventuelt med forskellige vilkårlige værdier.

Den første variant med nedarvning af konstante egenskaber er automatisk omfattet af den almindelige, simple nedarvning der er defineret for type-instance konceptet i det generelle systemmodelleringssprog. Konstante egenskaber kan uden videre modelleres som delsystemer med whole-part relationer i forhold til type systemet.

De to øvrige varianter med nedarvning af bundne men varierende egenskaber kan generaliseres til et med en antagelse om at alle elementer i en systemmodel er af typen "system". Dermed er varianten med nedarvning af varierende egenskaber med vilkårlige værdier et special tilfælde af varianten med nedarvning af varierende egenskaber med givne værdier.

En sådan nedarvning af varierende egenskaber er ikke umiddelbart omfattet af den almindelige, simple nedarvning der er defineret for type-instance konceptet i det generelle systemmodelleringssprog, men kræver supplerende sikring af konsistens og komplethed mellem typerne og forekomsterne i systemmodellen. Hvis en type har egenskaber i form af whole-part delsystemer der selv er (rene) type definitioner, skal alle forekomster af typen tilsvarende have egenskaber i form af whole-part delsystemer der tilsvarende er typer af typens delsystem typer for at systemmodellen er konsistent og komplet.

Definition af type-instance nedarvning konsistens og komplethed kriterie

$\forall$ type, forekomst, egenskabtype, egenskabsubtype, egenskabdelsystem

• Type-Instance(type, forekomst) $\wedge$ Whole-Part(type, egenskabtype) $\wedge$ Type-Instance(egenskabtype, egenskabsubtype) $\wedge \neg$Whole-Part(egenskabtype, egenskabdelsystem)

$\rightarrow \exists$ Whole-Part(forekomst, egenskabsubtype) $\wedge$ Type-Instance(egenskabtype, egenskabsubtype)

Whole-part koncept

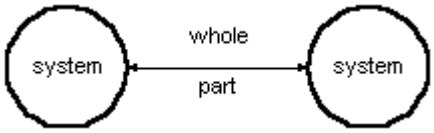
prev next up

sNets [1998] sNet Formalism - Technical Report, The

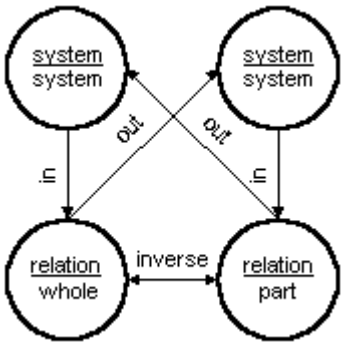
Ethvert system består konceptuelt (ikke nødvendigvis fysisk) af et antal delsystemer og relationer, der i sig selv er systemer. Ethvert system kan ligeledes indgå som delsystemer og relationer i et antal andre systemer i forbindelse med overlappende systemer og forskellige synsvinkler. Whole-part relationer er gensidigt definerende og inverterede.

Whole-part relationer kan anvendes til definering og modellering af forskellige abstraktionsniveauer for opdeling og detaljering af systemer i delsystemer; samt til gruppering og separering af (typisk) tæt relaterede delsystemer.

Semantik for whole-part koncept



Definition af whole-part koncept



Komposition af systemer fra delsystemer sker over flere niveauer mellem alle overliggende og underliggende systemer i whole-part relation hierarkiet inklusiv nedarvede relationer fra type-instance hierarkiet, således at der principielt ikke er nogen forskel på om et system direkte er en del af et andet system eller indirekte er en del af et andet system via et antal mellemliggende niveauer. Niveauerne i et whole-part relation hierarki udgør i systemmodellern kun forskellige abstraktionsniveauer og dermed ikke nødvendigvis forskellige absolutte, strukturelle, "fysiske" niveauer. Der kan således frit indsættes og fjernes mellemniveauer i whole-part relation hierarkier.

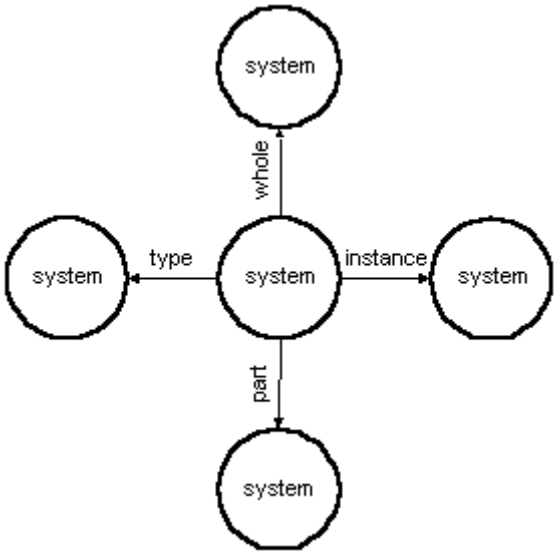
Definition af whole-part komposition
∀ system, delsystem, deldelsystem
• Whole-Part(system, delsystem) ∧ Whole-Part(delsystem, deldelsystem)
→ Whole-Part(system. deldelsystem)
∀ type, forekomst, egenskab
• Type-Instance(type, forekomst) ∧ Whole-Part(type, egenskab)
→ Whole-Part(forekomst, egenskab)

Systemhierarki systemmodel

prev next up

Modelleringssprogets bundne systemmodel består udelukkende af whole, part, type og instance relationer (som dog kan have 0-kardinalitet således at alle disse relationer i praksis kan udelades for de enkelte (del)systemer). Disse relationer udgør en todimensionel systemhierarki systemmodel med henholdsvis et type-instance og whole-part hierarkier.

Semantik for systemhierarki systemmodel



De fleste systemer vil desuden modelleres med navngivning af systemet, delsystemerne og relationerne. Dette er dog ikke en forudsætning, og det betragtes her ikke som en egentlig egenskab ved et system. Navngivning knyttet til systemmodelleringen betragtes her som meta-egenskaber i forbindelse med et generelt systemmodelleringsværktøj på linie med for eksempel angivelse af præsentationsmåde og adgangsrettigheder for de enkelte systemer.

Implementering

prev next up

Zeigler [1976] Theory of Modelling and Simulation

Jensen [1997] A Brief Introduction to Coloured Petri Nets

Jensen [1994] An Introduction to the Theoretical Aspects of Coloured Petri Nets

Det her formulerede generelle systemmodelleringssprog i form af semantiske system-relation, type-instance og whole-part koncepter giver mulighed for systemmodellering af vilkårlige systemer. Systemmodelleringen omfatter dog kun strukturelle og abstrakte elementer for systemet.

I forbindelse med generel systemmodellering af computerbaserede systemer er der desuden behov for at kunne modellere konkrete, eksekverbare systemmodeller i form af programkode for en bestemt computer. Dermed skal systemmodellen resultere i sekvenser af bits og bytes eller tilsvarende data, der repræsenterer computerens sprog.

Det generelle systemmodelleringssprog omfatter ikke direkte sådanne konkrete data, men det er muligt at definere og modellere med for eksempel bits og bytes som generelle systemelementer. Disse har dog ikke umiddelbart nogen speciel betydning i forhold til andre generelle systemelementer.

Generelle systemmodelleringsværktøjer til systemmodellering af computerbaserede systemer bør derfor omfatte et antal aksiomatiske systemdefinitioner med speciel betydning i forhold til det generelle systemmodelleringsværktøj til for eksempel generering af computer filer med data og programmer.

Sådanne aksiomatiske systemdefinitioner kan for eksempel blot bestå af en "0 bit" og en "1 bit" hvormed bytes og alle andre computerbaserede data kan modelleres som generelle systemmodeller. Det kan eventuelt også være hensigtsmæssigt at definere "byte" og andre computerbaserede data som aksiomatiske systemer for at gøre det generelle systemmodelleringsværktøj mere effektivt og hensigtsmæssigt at anvende.

Udover aksiomatiske systemdefinitioner for bits til systemmodellering af computerbaserede systemer kan generelle systemmodelleringsværktøjer eventuelt også omfatte andre aksiomatiske systemdefinitioner for andre typer af systemmodeller som for eksempel DEVS og Petri Net med henblik på umiddelbar simulering. Det vil eventuelt også være muligt at integrere de forskellige aksiomatiske systemdefinitioner, således at én generel systemmodel umiddelbart kan anvendes til for eksempel både simulering og generering af direkte eksekverbar computer programkode.

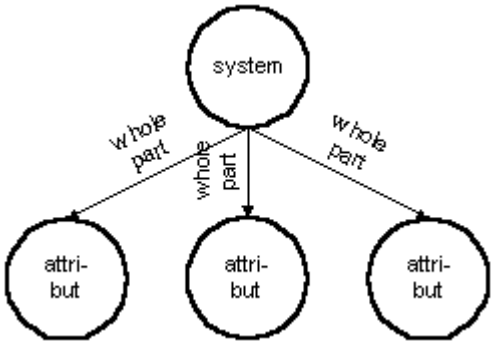
Datamodellering

prev next up

Datamodellering i forbindelse med computerbaserede systemer sker typisk i form af entitet-relation diagrammering med modellering af entiteter med tilknyttede attributter og nøgler samt kardinalitetsrelationer mellem de enkelte entiteter/attributter.

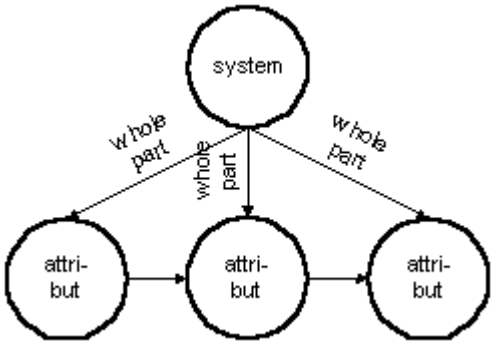
Med generel systemmodellering er modellering af entitet-relation modeller naturligvis også muligt men kun i det omfang at der specifikt ønskes entitet-relation modellering. Generel datamodellering laves med generelle whole-part relationer mellem systemer og delsystemer.

Semantik for sæt datamodellering



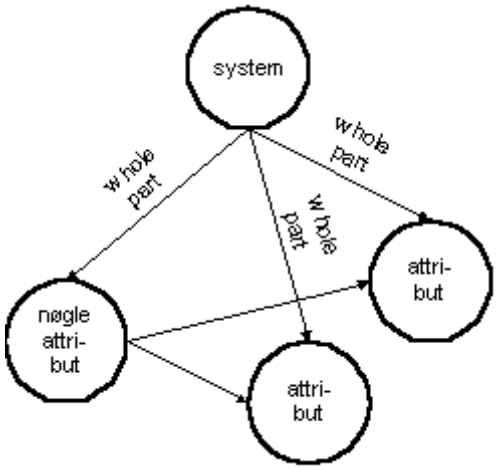
Ved ordnede data modelleres ordenen med generelle "fører-til" / "kommer-før" relationer mellem dem.

Semantik for sekvens datamodellering



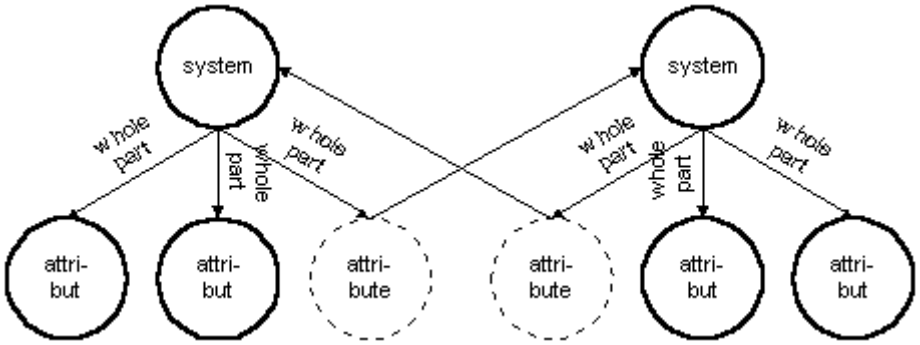
Determinerende data som for eksempel nøgleattributter for entiteter modelleres dette ligeledes med generelle "fører-til" / "kommer-før" relationer mellem de determinerende data og de tilknyttede/afledte data.

Semantik for nøgleattribut datamodellering



Datamodellering af entitet-relation relationer mellem entiteter modelleres med generelle whole-part relationer ved at modellere relaterede entiteter som attributter / delsystemer. Mange-til-mange entitet-relation relationer modelleres således som cirkulære whole-part relationer.

Semantik for mange-til-mange entitet-relation datamodellering.



Generel datamodellering med generel systemmodellering ligner således "gammeldags" hierarkiske databaser og netværksdatabaser. Dette gælder dog kun modelleringsmæssigt men dermed ikke nødvendigvis implementeringsmæssigt. Entitet-relation modellering kan modelleres eksplicit for specifikke systemer eller kan modelleres som generelle koncepter på samme måde som whole-part relationer, der udelukkende er defineret med det generelle modelleringssprogs kerne system-relation koncept. Det er desuden muligt at transformere mellem whole-part og entitet-relation datastrukturer.

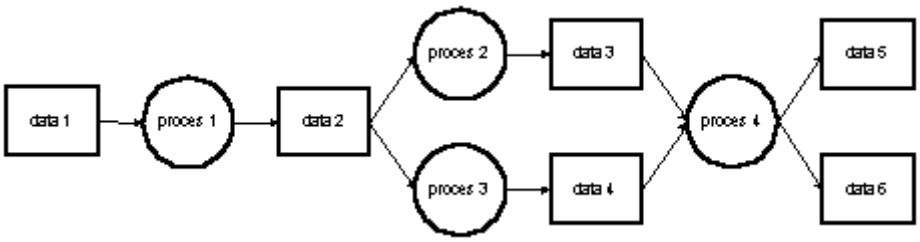
Procesmodellering

prev up

Procesmodellering i forbindelse med computerbaserede systemer sker typisk på meget forskellige måder i forbindelse med forskellige systemudviklingsmetodologier samt aktiviter og abstraktionsniveauer. Der er dog visse fællestræk, og specielt kan der laves fællestræk igennem generalisering af koncepterne.

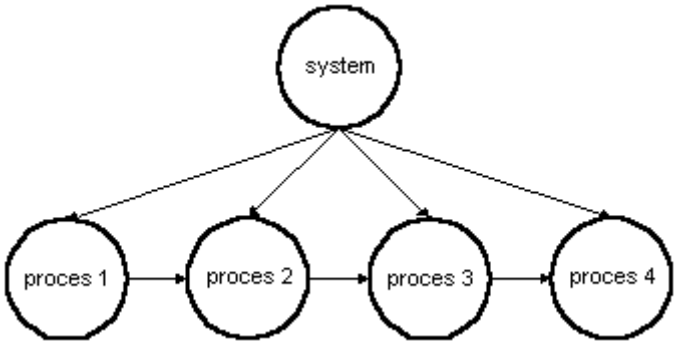
På højere abstraktionsniveauer i forbindelse med analyse og design modelleres processer som delprocesser forbundet med relationer i form af data. I forbindelse med implementering modelleres processer derimod som sekvenser af programinstruktioner. Disse sekvenser af programinstruktioner kan dog ligeledes transformeres til netværk af programinstruktioner forbundet med relationer i form af de data, som den enkelte programinstruktion benytter. Sådanne transformationer er almindelige i moderne kompilere i forbindelse med registerallokering for og optimering af programkode. Processer kan således logisk set generelt modelleres som netværk af delprocesser og data.

Logisk procesmodellering



I forbindelse med computerbaseret eksekvering skal processer dog nødvendigvis transformeres til sekvenser af maskinkode programinstruktioner. Data eksisterer kun indirekte igennem computerens lagerenheder og CPUens interne registre.

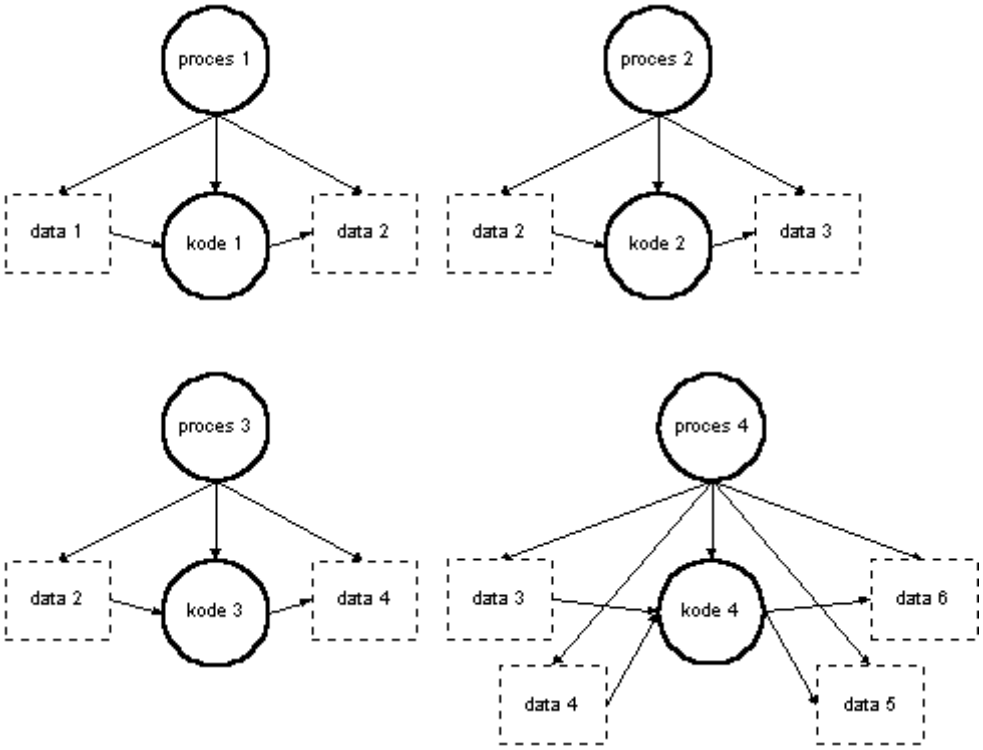
Implementering af procesmodellering



Internt i computerens CPU kan sekvenserne af maskinkode programinstruktioner dog også logisk set modelleres som netværk af maskinkodeinstruktioner forbundet med relationer i form af de registre, som den enkelte maskinkodeinstruktion benytter.

Ved at bringe procesmodelleringen et niveau dybere fra programmodellering til modellering af computerens CPU kan der bevares én form for procesmodellering som modellering af netværk af processer og data, der er konsistent med modellering af statiske programstrukturer som sekvenser af programinstruktioner.

Modellering af programinstruktioner



Al procesmodellering fra uformel højniveau analyse til formel lavniveau implementering kan således laves på samme måde i form af netværk af processer og data.

Konklusion på generel systemmodellering

prev up level

Systemvidenskab

Computerbaserede systemer

Generelt systemmodelleringssprog

Selvom generel systemteori ikke umiddelbart kan tilbyde et integreret systemparadigme og et systemmodelleringssprog der er direkte anvendeligt til systemudvikling af computerbaserede systemer, er generel systemteori i vid udstrækning relevant og interessant i forbindelse med systemudvikling af computerbaserede systemer.

Generel systemteori omfatter en række forskellige teorier for grundlæggende koncepter for systemer samt systemmodellering og systemanalyse, og generel systemteoris fokus på organiseret kompleksitet karakteriserer i vid udstrækning også computerbaserede systemer. Generel systemteoris store bredde fra hårde systemteorier for maskinelle systemer til bløde og kritiske systemteorier for sociale systemer karakteriserer ligeledes også computerbaserede systemer omfattende såvel de tekniske delsystemer som de omgivende sociale delsystemer, samt selve systemudviklingsprocessen.

Håndtering og simplificering af systemkompleksitet er centralt for såvel generel systemteori som systemudvikling af computerbaserede systemer, og generelle principper som ekskludering og gruppering/klassificering af variable samt opdeling af systemer i delsystemer er relevante for enhver formulering af systemmodelleringssprog og systemudviklingsmetoder.

Klassificering og karakterisering af forskellige systemtyper med forskellige systemiske egenskaber er desuden et middel til rigere sprogbrug og øget viden om systemer samt til mere meningsfyldte og effektive systembeskrivelser, hvilket nu også er under hastig udbredelse i forbindelse med systemudvikling af computerbaserede systemer i form af mønstre.

Endelig omfatter generel systemteori en række forskellige generelle systemmodelleringssprog til systemmodellering af dynamiske systemer, og selvom de ikke umiddelbart er orienterede mod og velegnede til direkte implementering af computerbaserede systemer, er de anvendte principper og metoder i vid udstrækning relevante for formuleringen af et generelt systemmodelleringssprog orienteret mod systemudvikling af computerbaserede systemer. Generel systemteori omfatter desuden en række forskellige teorier og metoder for bearbejdning og analyser af generelle systemmodeller der også i vid udstrækning er anvendelige i forbindelse med systemmodeller for computerbaserede systemer.

Med udgangspunkt i grundlæggende koncepter fra generel systemteori er der her formuleret et generelt systemmodelleringssprog orienteret mod systemudvikling af computerbaserede systemer. Det generelle systemmodelleringssprog omfatter en formel semantik for nogle få, simple koncepter der er tilstrækkelige til systemmodellering af komplekse systemer der omfatter såvel uformelle, sociale delsystemer som formelle, maskinelle delsystemer i form af direkte eksekverbar computer programkode, og det vises hvordan det generelle systemmodelleringssprog kan anvendes til systemmodellering af computerbaserede data og processer.

Systemudvikling med generel systemmodellering

next up level

Lagring af generelle systemmodeller

Generelle præsentationsværktøjer

Generelle analyseværktøjer

Generelle vs. specialiserede værktøjer

Generel systemmodellering systemudviklingsproces

Det her specificerede generelle systemmodelleringssprog beskriver udelukkende modelleringssprogets semantiske koncepter uden krav til og retningslinier for syntaks for hvordan konkrete modeller skal/kan repræsenteres visuelt og "fysisk"/digitalt, samt hvordan systemmodeller udarbejdes i forbindelse med et systemudviklingsforløb. Forslag til hvordan dette kan foregå beskrives nærmere i de følgende afsnit.

For at muliggøre udveksling af generelle systemmodeller mellem systemudviklere samt systemudviklingsværktøjer uden speciel/manuel bearbejdning er specifikation af formater for lagring af generelle systemmodeller i filer og/eller databaser nødvendig. En standard for lagring af generelle systemmodeller kan desuden ske igennem specifikation og implementering af grænseflader til services med bagvedliggende (eventuelt ikke-standardiserede) lagringsformater. Lagring af generelle systemmodeller kan eventuelt mest hensigtsmæssigt ske ved hjælp af eksisterende standarder for filer, databaser og services.

For effektiv konstruktion og analyse af generelle systemmodeller er letlæselige og letskrivelige repræsentationer nødvendige for systemudviklerne. Sådanne repræsentationer behøver ikke nødvendigvis at være standardiserede når blot de er i overensstemmelse med det generelle modelleringssprogs semantik. Forskellige former for repræsentationer fremhæver forskellige aspekter ved de generelle systemmodeller, og anvendelse af flere forskellige repræsentationer kan således være hensigtsmæssige for systemudviklere. Systemudviklere har mulighed for at benytte hver deres foretrukne/foreskrevne repræsentationer, der er bedste egnede for netop dem og deres systemudviklingsaktiviteter. For kommunikation mellem systemudviklere om generelle systemmodeller (herunder uddannelse i generel systemmodellering) vil nogle fælles standarder for generelle repræsentationer dog formodentligt være hensigtsmæssige/nødvendige. Generelle systemmodeller kan passende præsenteres som en række mere eller mindre kendte og udbredte modelleringsformater.

Som hjælp til modellering og vurdering af generelle systemmodeller kan generelle analyseværktøjer være nyttige. Analyseværktøjer kan analysere generelle systemmodeller for forskellige karakteristika og eventuelt foreslå forbedringer i form af omstruktureringer samt validere generelle systemmodeller udfra forskellige kriterier. Herunder vil generel systemmodellering under visse forudsætninger også umiddelbart kunne anvendes til formel specifikation og validering samt automatisk generering af formelle specifikationer til verifikation. Umiddelbart er nogle kendte og udbredte analysemetoder anvendelige i forbindelse med generel systemmodellering og videre studier og udvikling vil formodentligt resultere i yderligere generelt anvendelige metoder samt mere specialiserede metoder.

Et komplet systemudviklingsforløb kan principielt gennemføres udelukkende med generel systemmodellering og generelle systemmodelleringsværktøjer, men for gradvis inddragelse af generel systemmodellering i eksisterende systemudviklingsorganisationer er integration af generel systemmodellering med eksisterende, traditionelle systemudviklingsværktøjer nødvendig. Anvendelse af specialiserede systemudviklingsværktøjer (med generel systemmodellering som fundament) vil desuden være hensigtsmæssig for effektiv gennemførelse af de enkelte systemudviklingsaktiviteter fremfor udelukkende at anvende generelle systemudviklingsværktøjer.

Muligheden for én integreret systemmodel igennem hele systemudviklingsforløbet med generel systemmodellering og generelle systemudviklingsværktøjer betyder at selve systemmodellen og systemmodelleringsværktøjerne ikke lægger nogen begrænsninger på systemudviklingsprocessen. Systemudviklingsprocessen kan uden videre frit sammensættes af de bedste og foretrukne systemudviklingsaktiviteter, -metoder og -værktøjer for den enkelte systemudviklingsopgave og systemudvikler. Desuden kan nye og eksperimentielle systemudviklingsaktiviteter, -metoder og -værktøjer uden videre frit inddrages i systemudviklingsforløbet.

Lagring af generelle systemmodeller

next up

CDIF Technical Committee [1994] CDIF Framework for Modeling and Extensibility

Date & Darwen [1997] A Guide to the SQL Standard

Howes, Smith & Good [1999] Understanding and Deploying LDAP Directory Services

Object Management Group [1997] Meta Object Facility

Standarder for lagring af generelle systemmodeller er nødvendige for udveksling og deling af systemmodeller mellem forskellige systemudviklere indenfor det enkelte systemudviklingsforløb; mellem forskellige systemudviklingsforløb for (dele af) systemmodeller som genbrugskomponenter; samt mellem forskellige systemudviklingsorganisationer for (dele af) systemmodeller som standardkomponenter.

Forskellige former for lagring af generelle systemmodeller er nødvendige i forskellige kontekster. Filer muliggør simpel men begrænset udveksling og deling af generelle systemmodeller mellem forskellige systemudviklere og systemudviklingsværktøjer. Relationsdatabaser muliggør kontrolleret deling og administration af generelle systemmodeller mellem forskellige systemudviklere og systemudviklingsværktøjer samt integration med eventuelle andre organisatoriske systemer. Distribuerede databaser muliggør kontrolleret deling og administration af generelle systemmodeller mellem forskellige systemudviklingsorganisationer. Distribuerede objektarkitekturer muliggør deling af generelle systemmodeller og systemudviklingsværktøjer.

Formatet for generelle systemmodeller er vilkårligt blot det understøtter de få, simple grundlæggende elementer i generel systemmodellering i form af udefinerede "ting" bestående af digitale data samt simple whole/part og type/subtype relationer mellem disse. For at lette implementering af generel systemmodellering samt integration og udnyttelse af eventuelle eksisterende værktøjer kan det være hensigtsmæssigt at anvende eksisterende standarder til lagring af generelle systemmodeller.

CDIF (CASE Data Interchange Format) er et sæt standarder for udvekslings af modeller mellem CASE værktøjer. CDIF koordineres med og foreslåes som standard for fil-baseret udveksling af MOF (Meta Object Facility) og UML (Unified Modelling Language) systemmodeller. CDIF kunne således også være oplagt som format for generelle systemmodeller af hensyn til eventuel integration med eksisterende CASE værktøjer selvom CDIF umiddelbart virker forholdsvis omfattende og kompliceret i forhold til hvad de få og simple koncepter som generelle systemmodeller gør brug af.

CASE værktøjer lagrer i vid udstrækning systemmodeller i relationsdatabase-baserede "repositories" og relationsdatabaser er desuden de facto standard for databaser generelt. Generelle systemmodeller vil let kunne lagres i en simpel relationsdatabase i form af for eksempel en enkelt tabel med alle relationer og en enkelt tabel med alle (del)systemers dataværdier. Dermed vil systemmodeller relativt let kunne integreres med andre modeller og systemer samt understøttes af et bredt udvalg af værktøjer til analyse, bearbejdning og administration.

Relationsdatabaser er primært beregnet/anvendt til intraorganisatoriske databaser og kun begrænset egnede til interorganisatoriske databaser af sikkerhedsmæssige og administrative årsager. X.500 og LDAP (Lightweight Directory Access Protocol) angiver derimod standarder for distribuerede, hierarkiske databaser som kan være hensigtsmæssige for deling af generelle systemmodeller på tværs af organisationer. Lagring af generelle systemmodeller i X.500/LDAP databaser vil blot kræve definition af et par X.500/LDAP objektklasser for generelle systemmodel relationer og systemer/"ting".

MOF (Meta Object Facility) er et sæt standarder for deling af systemmodeller samt deling og integration af systemmodelleringsværktøjer i form af distribuerede netværksservices. Generelle systemmodeller vil forholdsvist let kunne modelleres igennem MOF og generelle systemmodelleringsværktøjer vil ligeledes kunne implementeres som MOF services.

Generelle præsentationsværktøjer

prev next up down

Modelstrukturer

Præsentationsformer

Der er ingen direkte bundne/indbyggede præsentationer af generelle systemmodeller idet disse i hvert fald set som generelle systemmodeller kun omfatter semantiske strukturer og udefinerede dataværdier. I det omfang som systemmodellerne omfatter egenskaber for præsentation vil disse være systemspecifikke og ikke umiddelbart anvendelige for et generelt præsentationsværktøj. En generel systemmodel kan for eksempel være en model af et grafisk filformat indeholdende en kompliceret tegning lavet med et bestemt grafisk tegneværktøj, men for et generelt systemmodelleringsværktøj og præsentationsværktøj vil denne model udelukkende bestå af for eksempel en hierarkisk struktur af koder for grafiske elementer.

Generelle præsentationsværktøjer kan således umiddelbart kun benyttes til at præsentere generelle systemmodeller udfra forskellige generiske strukturer og præsentationsformer. Derudover kan generelle systemmodelleringsværktøjer og præsentationsværktøjer bruges til at definere specielle præsentationsmodeller for de enkelte generelle systemmodeller.

I forbindelse med modellering af komplekse systemer er det vigtigt at kunne fokusere på forskellige, specifikke aspekter af systemmodellen ad gangen for at kunne håndtere kompleksiteten. For generelle systemmodeller omfatter relevante aspekter strukturerne af de grundlæggende whole/part og type/subtype relationer mellem de forskellige delsystemer samt relationer mellem delsystemerne indenfor et enkelt (del)system. Derudover kan et vilkårligt antal forskellige strukturer bestående af forskellige systemmodel specifikke relationer og (del)systemer være relevante. Generelle præsentationsværktøjer bør kunne præsentere sådanne forskellige aspekter ved generelle systemmodeller i form af forskellige strukturer for systemudviklerne.

Til præsentation af forskellige generelle systemmodel strukturer kan en række forskellige visualiseringsformer, der hver især fremhæver forskellige aspekter ved strukturerne, benyttes. Anvendelige formater omfatter tekstuelle og matematiske/logiske beskrivelser; tabeller og matricer; hierarkiske diagrammer og netværksdiagrammer; samt higraph og 3D modeller.

Modelstrukturer

next up

Consens, Mendelzon & Ryman [1992] Visualizing and Querying Software Structures

Kleyn & Browne [1993] A High Level Language for Specifying Graph Based Languages and their Programming Environments

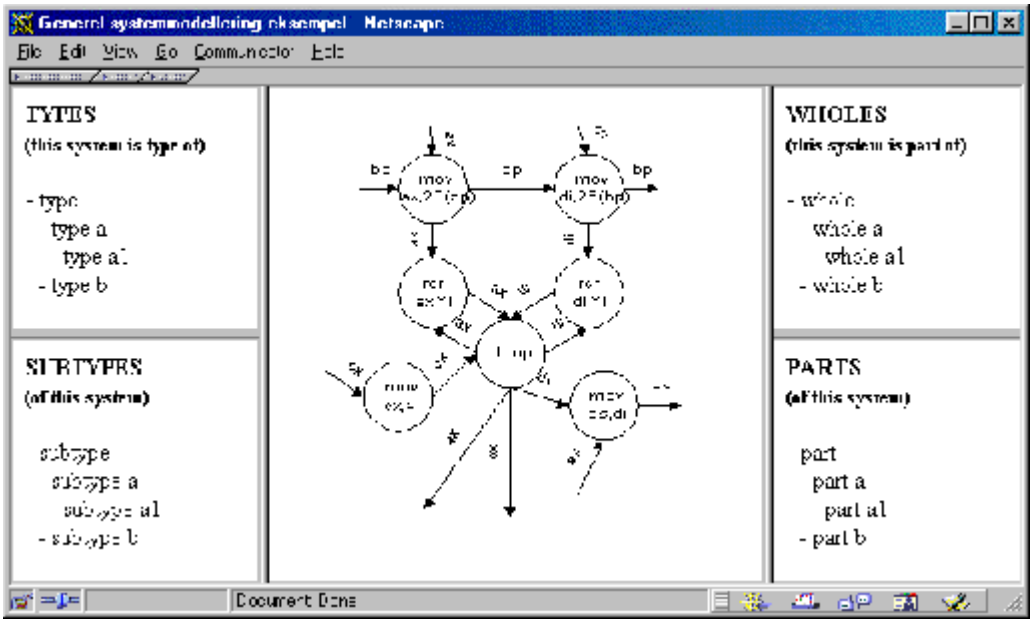
Med whole-part og type-instance relationer som grundlæggende elementer i generel systemmodellering er præsentation af disse strukturer central. Disse relationer er ligeledes naturlige for navigering "op" og "ned" i generelle systemmodeller i forbindelse med generelle systemmodelleringsværktøjer.

Whole-part relationer udgør multidimensionelle træstrukturer for generelle systemmodeller, der svarer til mange traditionelle systemudviklingsmetoders strukturer for systemmodeller som for eksempel organisationsdiagrammer, funktionsdiagrammer, hierarkiske dataflowdiagrammer, strukturdiagrammer. Whole-part strukturer afspejler ligeledes "naturlige", ikke-normaliserede datastrukturer men adskiller sig dermed fra traditionelle systemudvikling datastrukturer i form af entitets-relationsstrukturer. (Entitets-relationsstrukturer kan dog automatisk udledes af whole-part strukturer.)

Type-instance relationer udgør ligeledes multidimensionelle træstrukturer for generelle systemmodeller, der svarer til traditionelle systemudviklingsmetoders strukturer for systemmodeller som for eksempel klassifikationer samt type og objektklasse hierarkier.

Derudover omfatter de grundlæggende elementer i generel systemmodellering kun simple relationer mellem delsystemer indenfor et system, der udgør et netværk af delsystemerne og relationerne. Disse svarer til mange andre systemmodelleringsstrukturer, som for eksempel dataflowdigrammer og andre former for flowdiagrammer.

Et basalt, generelt systemmodelleringsværktøj kunne dermed være opbygget således:



Generelle systemmodeller kan typisk blive meget omfattende og komplekse med mange forskellige slags delsystemer og specielle relationer i form af delsystemer der forbinder andre delsystemer. For at håndtere kompleksiteten og fokusere på bestemt aspekter ved systemodellerne vil det være hensigtsmæssigt at skjule eller ignorere bestemte delsystemer og relationer for at fremhæve de resterende.

Ved definition af en præsentation af en generel systemmodel bør det således være muligt at definere hvilke delsystemer og typer af delsystemer præsentationen skal omfatte. Ved systemanalyse og -design kan programinstruktioner for eksempel eventuelt ignoreres og omvendt kan alt andet end programinstruktioner eventuelt ignoreres ved teknisk programdokumentation.

Derudover bør der ved definition af en præsentation af en generel systemmodel ligeledes

være mulighed for at definere om de enkelte delsystemer og typer af delsystemer skal præsenteres som delsystemer eller skal "foldes sammen" og "skjules" i form af relationer. Ved dataflowdiagrammering kan "data" delsystemer for eksempel skjules således at de kun medtages som relationer mellem "proces" delsystemer; og ved tilstandsdiagrammering kan "data" og "proces" delsystemer for eksempel skjules således at de kun medtages som relationer mellem "tilstand" delsystemer.

Ved at kunne henholdsvis ignorere og skjule forskellige delsystemer og typer af delsystemer samt at kunne begrænse bredden og dybden af modelstrukturer i forbindelse med præsentation af generelle systemmodeller er der stor fleksibilitet til at præsentere forskellige typer af og aspekter ved generelle systemmodeller.

Valget af relevante modelstrukturer til en præsentation af en generel systemmodel kan ske ved simpel udvælgelse/fravælgelse af de enkelte elementer og typer af elementer ved simple og mindre generelle systemmodeller, men ved komplekse og større generelle systemmodeller er dette dog ikke hensigtsmæssigt. Ved komplekse og større generelle systemmodeller er der behov for at kunne formulere komplekse forespørgsler der definerer udsnit af generelle systemmodeller. GLIDE og GraphLog er eksempler på graf-baserede modelleringsmetoder, der omfatter et forespørgselsprog til definition af modelstruktur udsnit.

Med avancerede forespørgselssprog er det desuden muligt at definere komplekse forespørgsler med transformation af de hierarkiske generel systemmodel strukturer fra for eksempel dataflow grafer til objektorienterede grafer og fra hierarkiske datastrukturer til entitet-relation grafer.

Det er således muligt at opbygge komplekse systemmodeller med nogle få generelle strukturer som system-relation netværk samt whole-part og type-instance hierarkier, mens at det samtidigt er muligt at transformere sådanne systemmodeller til mere selektive og effektive repræsentationer af disse og afledte strukturer.

Præsentationsformer

prev up

Roman & Cox [1992] Program Visualization: the Art of Mapping Programs to Pictures

Harel [1998] On Visual Formalisms

Feijls & De Jong [1998] 3D Visualization of Software Architectures

Battista, Eades, Tamassia & Tollis [1999] Graph Drawing: Algorithms for the Visualization of Graphs

Selvom alle generelle systemmodeller udelukkende består af de få og simple grundlæggende elementer, kan de afspejle mange forskellige systemtyper, og ved hjælp af forskellige præsentationsformer kan forskellige aspekter ved systemmodellerne fremhæves.

Hver enkelt systemmodel kan præsenteres på mange forskellige måder men anvendeligheden anhænger af den enkelte systemtype og systemmodelleringsaktivitet. Alle generelle systemmodeller kan uden videre præsenteres med de følgende generelle præsentationsformer udfra de grundlæggende elementer.

Systemmodeller i form af rent verbale beskrivelser samt verbale beskrivelser i forbindelse med strukturelle systemmodeller er i vid udstrækning nødvendige, og sådanne verbale beskrivelser bør indgå i generelle systemmodeller sammen med de strukturelle modeller af hensyn til sikring af integritet i systemmodellerne. I det omfang at generelle systemmodeller indeholder tekstuelle data kan disse præsenteres i sekventiel, rapport/dokument form udfra deres whole-part relationer og indbyrdes (rækkefølge) relationer.

For kompakte og eksakte præsentationer af systemmodeller er matematisk/logiske beskrivelser hensigtsmæssige eventuelt med henblik på matematisk/logisk analyse af systemmodellerne. Semantikken for grundelementerne i generel systemmodellering kan specificeres formelt og ud fra dette kan generelle systemmodeller ligeledes formuleres som formelle specifikationer. Relevante former i forbindelse med computerbaserede systemer kunne være prædikativ logik og (Extended) Backus-Naus Form.

System S = System S.1 + System S.2 + System S.3 + Relation R.1(System S.1, System S.2) + Relation R.2(System S.1, System S.3) + Relation R.3(System S.2, System S.3)
System S.1 = ...
System S.2 = ...
System S.3 = ...
Relation R.1 = ...
Relation R.2 = ...
Relation R.3 = ...
.....

Præsentation af generelle systemmodeller i tabelform som rækker af "tuples" bestående af enkelte delsystemer og/eller relationer er anvendelige i forbindelse med relationsdatabase relateret bearbejdning af generelle systemmodeller samt i forbindelse med Petrinet modellering og simulering.

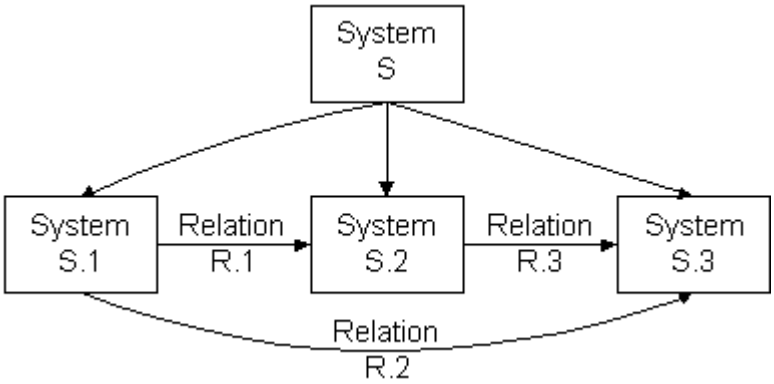
System S	whole/part	System S.1
System S	whole/part	System S.2
System S	whole/part	System S.3
System S.1	Relation R.1	System S.2
System S.1	Relation R.2	System S.3
System S.2	Relation R.3	System S.3
System S.1	whole/part	.....
System S.2	whole/part	.....

System S.3	whole/part	.....
Relation R.1	whole/part	.....
Relation R.2	whole/part	.....
Relation R.3	whole/part	.....
.....	.....	.....

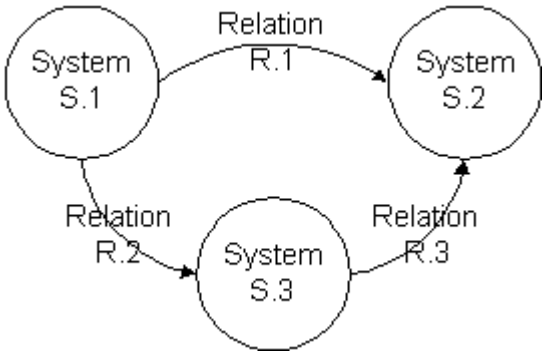
Overblik, grupperinger og mønstre for relationer mellem (del)systemer fremhæves ved præsentation af generelle systemmodeller i matriceform som er velegnet til matematiske graf analyse og transformation algoritmer. Sådanne udgør for eksempel udgangspunktet for gruppering af de tættest relaterede funktioner i programmoduler i forbindelse med Information Engineering CASE værktøjer.

	System S	System S.1	System S.2	System S.3	.....
System S		whole/part	whole/part	whole/part	
System S.1			Relation R.1	Relation R.2	
System S.2				Relation R.3	
System S.3					
.....					

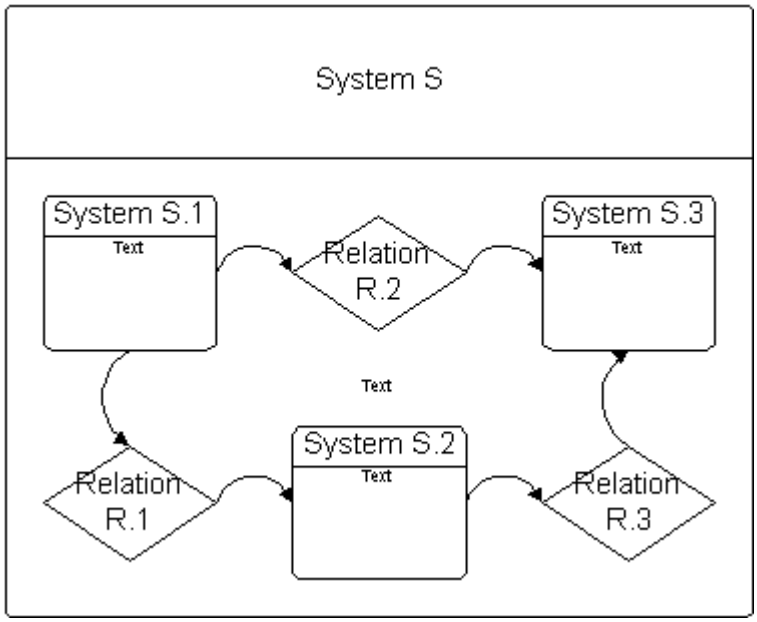
hierarkiske strukturer anvendes i vid udstræning i forbindelse med traditionel systemudvikling som for eksempel funktions-, struktur- og objektklassediagrammer og udgør ligeledes en del af de grundlæggende elementer i form af whole/part og type/instance relationer i generel systemmodellering. Hierarkiske præsentationsformer er således centrale og naturlige for præsentation af generelle systemmodeller.



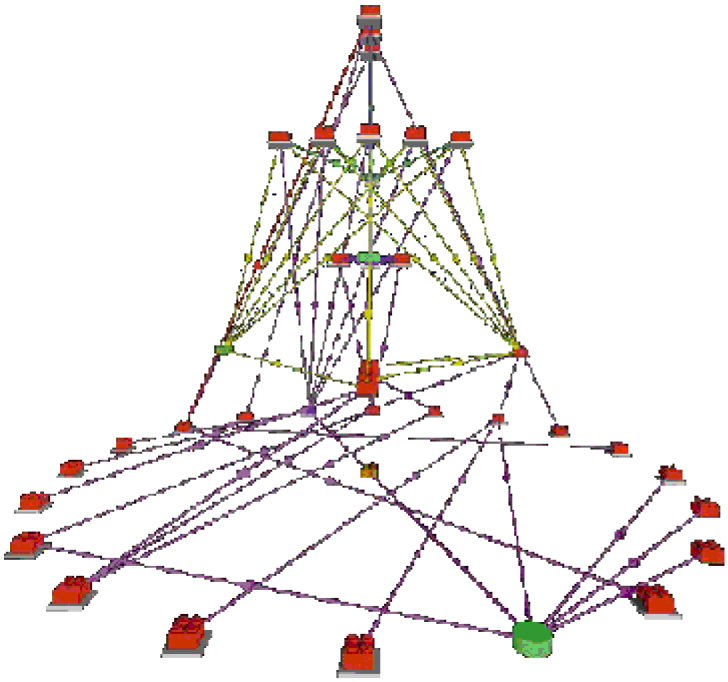
Netværk strukturer anvendes ligeledes i vid udstrækning i forbindelse med traditionel systemudvikling som for eksempel entitets-relations- og dataflowdiagrammer samt Petrinet og udgør ligeledes en del af de grundlæggende elementer i form af simple relationer i generel systemmodellering. Netværk præsentationsformer er således også centrale og naturlige for præsentation af generelle systemmodeller.



hierarkiske præsentationsformer og netværk præsentationsformer fremhæver forskellige strukturer og aspekter ved generelle systemmodeller men giver ikke en samlet præsentation. Topologiske præsentationsformer som higraphs omfatter såvel hierarkiske strukturer som netværk strukturer og muliggør således integreret præsentation af komplette generelle systemmodeller. (Dette eksempel viser en forholdsvis triviel higraph).



To-dimensionelle præsentationer har begrænset anvendelighed for store og komplekse generelle systemmodeller med mange delsystemer, mange tværgående relationer og flerdimensionelle hierarkiske strukturer, og opsplit af sådanne generelle systemmodeller i forskellige delpræsentationer er typisk nødvendigt. Tre-dimensionelle præsentationer af generelle systemmodeller giver bedre muligheder for præsentation af generelle systemmodeller med for eksempel netværk strukturer i den ene dimension og hierarkiske strukturer i den anden. Farve- og formkodning af forskellige typer af delsystemer og relationer giver yderligere muligheder for præsentation af flerdimensionelle hierarkiske strukturer i generelle systemmodeller.



Kilde: Feijls & De Jong [1998] 3D Visualization of Software Architectures

Med disse generelle pæsentationsformer er der gode muligheder for at visualisere forskellige aspekter ved generelle systemmodeller. For yderligere at udvide præsentationsmulighederne og knytte generel systemmodellering til eksisterende systemudviklingsmetoder og -værktøjer bør generelle præsentationsværktøjer desuden kunne variere udseendet af de enkelte elementer og typer af elementer i præsentationsformerne. De forskellige generelle præsentationsformer dækker en stor del af de anvendte, grundlæggende præsentationsformer i forbindelse med forskellige systemudviklingsmetoder, som dog benytter varierende notation. Med mulighed for at variere udseendet af de enkelte elementer og typer af elementer i de generelle præsentationsformer vil for eksempel generelle netværksdiagrammer kunne anvendes til såvel Rich Picture diagrammer som forskellige notationer for dataflow diagrammer samt Use Cases med passende notation.

Generelle analyseværktøjer

prev next up down

Sporbarhed

Bindinger og koblinger

Komplethed

Konsistens

Med generelle systemmodelleringsmodeller baseret på matematiske grafer som det her formulerede generelle systemmodelleringssprog er der mulighed for at udvikle en række generelle analyseværktøjer, der er anvendelige ved generel systemmodellering af vilkårlige systemer. I det følgende beskrives forskellige, grundlæggende generelle analyseværktøjer.

Ved traditionel systemudvikling med et antal forskellige systemmodeller for de forskellige systemudviklingsfaser og -aktiviteter er sporbarhed problematisk idet der ikke er direkte relationer imellem de forskellige systemmodeller. Ved generel systemmodellering med multidimensionelle systemmodeller er det derimod umiddelbart muligt at inddrage sporbarhed i systemmodellen og analysere og præsentere tværgående relationer mellem forskellige delsystemer.

Matematisk graf baseret cluster analyse af generelle systemmodeller er anvendelig som generelt analyseværktøj til vurdering og eventuel forbedring af strukturerer i generelle systemmodeller igennem kvantificering af bindinger i delsystemer og koblinger imellem delsystemer.

Ved generel systemmodellering med type-instance hierarkier hvor henholdsvis strukturelle krav for typer af delsystemer og basale systemelementer er defineret, kan disse anvendes til generel analyse af komplethed for generelle systemmodeller.

Generel systemmodellering med type-instance hierarkier med definition af strukturelle krav for typer af delsystemer og basale systemelementer, kan disse ligeledes anvendes til generel analyse af konsistens i generelle systemmodeller.

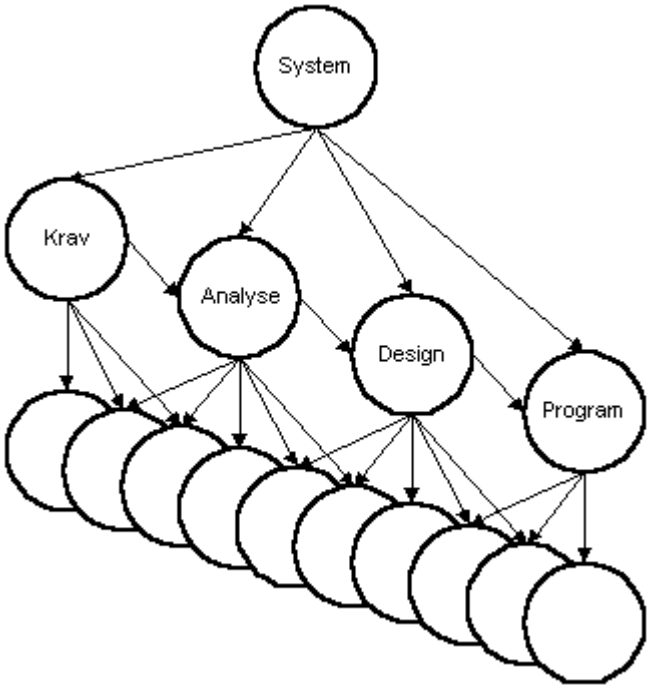
Sporbarhed

next up

Jarke [1998] Requirements Tracing

En af de væsentlige forskelle og fordele ved generel systemmodellering i forhold til traditionel systemmodellering er muligheden for at lave én integreret systemmodel for hele systemudviklingsprocessen. Det medfører også mulighed for at fastholde sporbarhed mellem de forskellige systemudviklingsaktiviteter og delsystemmodeller.

Ved generel systemmodellering i et systemudviklingsforløb med traditionelle systemudviklingsaktiviviteter og delsystemmodeller som kravspecifikation, systemanalyse, systemdesign og implementering kan disse modelleres som én hierarkisk generel systemmodel med systemanalyse underlagt kravspecifikation, systemdesign underlagt systemanalyse og implementering underlagt systemdesign; eller som én multidimensionel generel systemmodel med fælles delsystemer underlagt henholdsvis kravspecifikation, systemanalyse, systemdesign og implementering.



Med sådanne generelle systemmodel strukturer vil der umiddelbart være mulighed for sporbarhed mellem de forskellige dele af systemudviklingsaktiviteterne igennem de hierarkiske relationer i den generelle systemmodel. Det kræver dog naturligvis at systemudviklerne eller systemmodelleringsværktøjer sikrer sammenkædning mellem de forskellige delsystemer.

Bindinger og koblinger

prev next up

Yourdon [1989] Modern Structured Analysis

Wasserman & Faust [1994] Social Network Analysis: Methods and Applications

McHugh [1990] Algorithmic Graph Theory

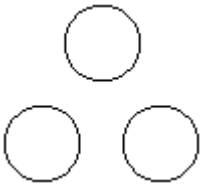
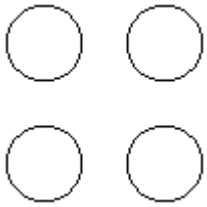
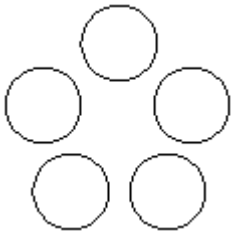
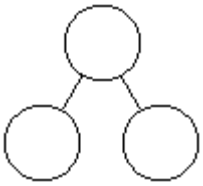
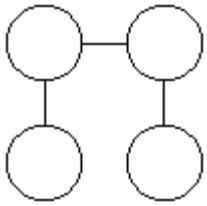
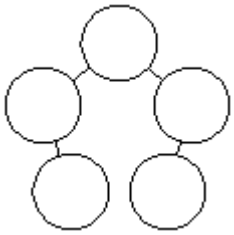
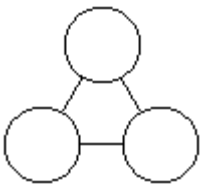
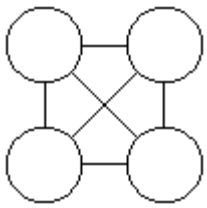
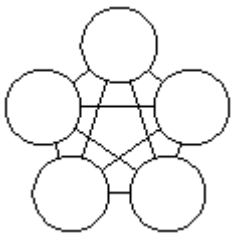
Butler, Esposito & Hebron [1999] Connecting the Design of Software to the Design of Work

To udbredte mål for kvaliteten af systemdesign og implementering er begreberne bindinger og koblinger, der anvendes i forbindelse med såvel objektorienteret som struktureret systemudvikling.

Bindingen karakteriserer den indbyrdes styrke i relationerne mellem delsystemerne indenfor et (del)system, hvor denne bør være så stærk som mulig, således at et (del)system udelukkende består af delsystemer med mange og tætte relationer.

Koblingen karakteriserer den indbyrdes styrke i relationerne mellem forskellige (del)systemer, hvor denne bør være så svag som mulig, således at relationerne mellem forskellige (del)systemer udelukkende består af få og løse/spredte relationer.

Med generel systemmodellering er det muligt at analysere bindinger og koblinger igennem graf cluster analyse, hvor styrken af relationerne mellem en given mængde delsystemer er antallet af relationer i forhold til antallet af delsystemer.

	3-node graf	4-node graf	5-node graf
Ingen relationer			
Svage relationer (sekvens)			
Stærke relationer (komplet graf)			

Generelle systemmodeller kan således umiddelbart analyseres for stærke koblinger i de hierarkiske relationer whole/part og type/instance relationer mellem delsystemerne samt for svage bindinger i de indbyrdes simple relationer mellem delsystemerne med henblik på eventuelt at forbedre den generelle systemmodel igennem omstruktureringer. Graf cluster analyse vil ligeledes kunne vise potentielt bedre grupperinger af delsystemerne i forbindelse med eventuelle omstruktureringer.

Komplethed

prev next up

Heimdahl & Leveson [1995] Completeness and Consistency Analysis of State-Based Requirements

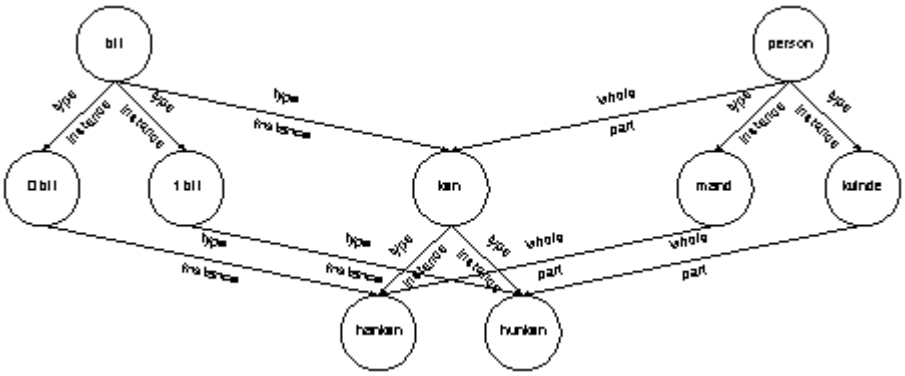
Kapur [1994] An Automated Tool for Analyzing Completeness of Equational Specifications

Generelle systemmodeller kan i sig selv principielt aldrig betegnes som værende komplette idet delsystemer i princippet altid vil kunne nedbrydes i yderligere delsystemer og altid vil kunne indgå i yderligere systemer. Omvendt kan generelle systemmodeller principielt altid betegnes som værende komplette idet de er konstruktivistiske modeller af subjektivt bestemte relevante aspekter ved et system.

Ved generel systemmodellering med anvendelse af typer/subtyper vil det dog være muligt at analysere om en given generel systemmodel mere konkret er komplet. Komplethed kan bestemmes udfra om et (del)system er i overensstemmelse med dets type(r) og om det er nedbrudt til et bestemt niveau i form af bestemte typer af delsystemer som for eksempel:

Krav til strukturen og indholdet af kravspecifikationer for en systemudviklingsorganisation kan være defineret som en generel systemmodel der benyttes som type for konkrete kravspecifikationer, der dermed kan analyseres for komplethed i form af om alle dele af kravspecifikationen er medtaget samt om alle dele af kravspecifikationen er nedbrudt til for eksempel en formelt eller uformelt defineret type af målbare udtryk.

Generelle systemmodeller for implementering af computerbaserede systemer kan analyseres for komplethed ved om alle delsystemer er nedbrudt til et niveau udelukkende bestående af elementer af en bestemt type af programkode instruktioner for en given computer eller binære data.



Krav til struktur og indhold for dokumentation af implementerede computerbaserede systemer for en systemudviklingsorganisation kan på tilsvarende måde som kravspecifikation eksemplet være defineret som en systemmodel der benyttes som type for konkrete implementeringer. Komplethed for computerbaserede system implementeringer kan dermed for eksempel analyseres udfra om alle programkode instruktioner er dokumenteret ved at indgå som dele af systemdesign og systemanalyse mønstre.

Med generel systemmodellering er der således fleksible, generelle muligheder for analysere generelle systemmodeller for komplethed igennem modellering og analyse af to forskellige generelle systemmodeller i form af henholdsvis en definition af kravene til struktur og indhold samt en konkret implementering.

Konsistens

prev up

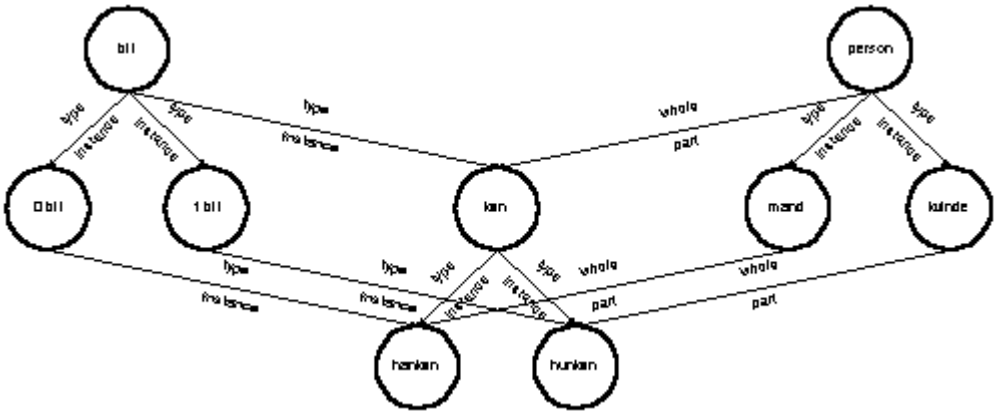
Heimdahl & Leveson [1995] Completeness and Consistency Analysis of State-Based Requirements

Heitmeyer, Jeffords & Labaw [1996] Automated Consistency Checking of Requirements Specifications

Generelle systemmodeller vil i sig selv altid være konsistente hvis de er i overensstemmelse med sematikken for det generelle modelleringsprog. Denne konsistens er dog ikke særlig interessant. Konsistens i generelle systemmodeller er interessant som konsistens mellem de forskellige niveauer af en generel systemmodel samt mellem de forskellige dimensioner af en multidimensionel generel systemmodel.

Med anvendelse af type-instance relationer vil det i et vist omfang være muligt at analysere om en given generel systemmodel er konsistent. Konsistens mellem et system og dets nedbrydning i delsystemer kan bestemmes udfra om systemet svarer til "summen" af de nedbrudte delsystemer således at de eksterne relationer på de to niveauer svarer til hinanden med hensyn til deres eksistens og deres typer. Ved multidimensionelle generelle systemmodeller vil en sådan konsistens ligeledes kræve/sikre overensstemmelse mellem de forskellige dimensioner, da forskellige dimensioner ikke kan have uforenelige typer af fælles delsystemer.

Ved generel systemmodellering af implementering af programkode instruktioner på forskellige niveauer vil konsistens analyse af delsystem typerne kunne afsløre uoverensstemmelser mellem datatyper og dermed mulige dataværdier, som for eksempel anvendelse af udefinerede talstørrelser på det overliggende niveau og anvendelse af 16 bit binære talstørrelser på det underliggende niveau. Konsistens analyse af uoverensstemmelser mellem eksistensen af eksterne relationer, hvor der forekommer eksterne relationer på et niveau der ikke forekommer på det andet, kan afsløre udefinerede/uhåndterede programtilstande, som for eksempel manglende håndtering af overløb/underløb ved addition/subtraktion af talværdier med et givent antal cifre.



Det er således muligt at lave detaljeret konsistens analyse af generelle systemmodeller men det kræver dog tilstrækkelig detaljeret anvendelse af type-instance relationer, hvor forskellige type-instance relationer nedbrydes til fælles typer af delsystemer som for eksempel bits. Ved anvendelse af uformelle/udefinerede type-instance relationer eller type-instance relationer uden fælles typer af delsystemer vil konsistens analyse kun kunne vise uoverensstemmelser mellem eksistensen af eksterne relationer på forskellige niveauer.

Generelle vs. specialiserede værktøjer

prev next up

Generelle systemmodelleringsværktøjer bør principielt kunne benyttes til systemmodellering for et komplet systemudviklingsforløb uden at det skulle være nødvendigt at benytte andre systemmodelleringsværktøjer. Dette vil dog eventuelt ikke være hensigtsmæssigt, da generelle systemmodelleringsværktøjer nødvendigvis netop må være meget generelle.

Generalitet og fleksibilitet opnåes i en vis grad på bekostning af udtryksfuldhed og effektivitet, og megen systemmodellering vil således formodentligt kunne laves lettere og måske bedre med specialiserede systemmodelleringsværktøjer end generelle systemmodelleringsværktøjer.

Generel systemmodellering bør således ikke ses som en erstatning for al anden systemmodellering i forbindelse med systemudviklingsforløb men derimod som et supplement eller en udbygning. Generel systemmodellering er tilstrækkelig til modellering af alle nødvendige strukturer i forbindelse med systemudviklingsforløb og kan dermed også anvendes som det underliggende modelleringssprog for specialiserede systemudviklingsværktøjer. Forskellige specialiserede systemudviklingsværktøjer samt generelle systemudviklingsværktøjer vil således kunne kombineres via den underligende fælles generelle systemmodellering.

Generel systemmodellering bør optimalt være det underliggende modelleringssprog for nye, specialiserede systemudviklingsværktøjer, men generel systemmodellering kan også anvendes sammen med eksisterende, specialiserede systemudviklingsværktøjer uden underliggende generel systemmodellering. Dette kan ske ved at definere generelle systemmodeller for de specialiserede fil formater og database skemaer, der anvendes af eksisterende, specialiserede systemudviklingsværktøjer, således at generelle systemmodelleringsværktøjer kan indlæse og generere (dele af) systemmodeller der er kompatible med de eksisterende, specialiserede systemudviklingsværktøjer.

Systemudvikling med generel systemmodellering kan således ske med generelle systemudviklingsværktøjer som de basale systemudviklingsværktøjer, der anvendes til generelle systemudviklingsaktiviteter og systemudviklingsaktiviteter hvortil der ikke findes bedre, specialiserede systemudviklingsværktøjer, men i det omfang at der findes bedre, specialiserede systemudviklingsværktøjer anvendes disse.

Generel systemmodellering systemudviklingsproces

prev up down

Systemudviklingsproces modellering

Organisation modellering

Problemsituation modellering

Kravspecifikation modellering

Systemanalyse modellering

Systemdesign modellering

Systemarkitektur modellering

Højniveau program modellering

Lavniveau program modellering

Generelle systemmodelleringsværktøjer baseret på det formulerede, generelle systemmodelleringssprog vil kunne anvendes til alle de mange forskellige systemmodelleringsaktiviteter i forbindelse med systemudviklingsforløb for systemudvikling af computerbaserede systemer. Alle systemmodelleringsaktiviteter kan ske med ét generelt systemmodelleringssprog og de forskellige systemmodeller for de forskellige systemudviklingsaktiviteter kan integreres i én generel systemmodel for det samlede system.

Systemudviklingsforløbet er dermed ikke bundet til givne systemudviklingsaktiviteter og givne relationer mellem disse af hensyn til systemmodelleringsværktøjerne og systemmodelleringen. Systemudviklingsforløbet kan derimod frit sammensættes og ændres udfra de aktuelle behov for det enkelte systemudviklingsforløb og de enkelte systemudviklere.

Systemudviklingsaktiviteterne kan være en vilkårlig sammensætning af forskellige systemmodelleringsaktiviteter med analytisk nedbrydning af overordnede, uformelle delsystemer til detaljerede, formelle delsystemer og konstruktivistisk samt konstruktivistisk opbygning af større, sammenhængende systemer udfra mindre, enkeltstående delsystemer.

I det følgende illustreres hvordan nogle forskellige, typiske systemudviklingsakvitiviteter kan understøttes af generel systemmodellering. Systemudviklingsaktiviteterne beskrives her i traditionel, "naturlig" rækkefølge uden at dette dog indikerer at dette nødvendigvis er tilfældet i et systemudviklingsforløb.

Indledningsvis anvendes generel systemmodellering til at modellere systemudviklingsforløbet selv med definition af de enkelte systemudviklingsaktiviteter og produkter samt deres indbyrdes relationer.

Systemets omgivelser i bred forstand modelleres i form af den omgivende organisation med henblik på systemudviklernes forståelse af domænet for systemet der eventuelt skal udvikles.

Den konkrete problemsituation, der har givet anledning til systemudviklingsforløbet, modelleres med henblik på detaljeret, gensidig forståelse af systemet og dets umiddelbare omgivelser for såvel involverede aktører som systemudviklere.

De specifikke krav til systemet der skal implementeres som et computerbaseret system modelleres med henblik på indgåelse af en gensidig aftale mellem kunden og systemudviklerne.

Systemets processer og data analyseres detaljeret med henblik på detaljeret forståelse af fænomener og relationer, der skal understøttes af de computerbaserede system.

En generel løsning designes for systemet med specifikation af data og processer samt deres indbyrdes relationer.

Den overordnede systemarkitektur for løsningen modelleres som et antal overordnede delsystemer, og den fysiske computer- og kommunikationsplatform for implementering af delsystemerne fastlægges.

Løsningen implementeres med detaljeret specifikation af computerbaserede algoritmer og datastrukturer samt grænseflader for systemet.

Endeligt detaljeres løsningen til lavniveau programinstruktioner og datarepræsentationer, der umiddelbart er eksekverbare med den valgte computer- og kommunikationsplatform.

Systemudviklingsproces modellering

next up

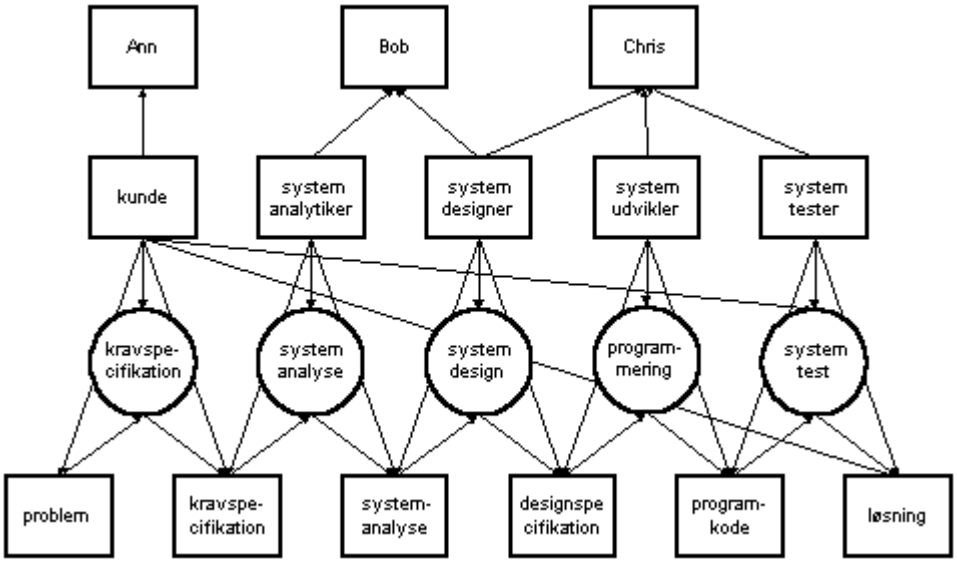
Curtis, Kellner & Over [1992] Process Modeling

Baldwin & Chung [1994] Design Methodology Management Using Graph Grammars

En af de væsentlige fordele ved generel systemmodellering er at den med mulighed for én integreret generel systemmodel er uafhængig af bestemte relationer mellem de forskellige traditionelle systemudviklingsaktiviteter, og dermed umiddelbart kan anvendes i forbindelse med vilkårlige systemudviklingsprocesser.

Et naturligt udgangspunkt for et systemudviklingsforløb med generel systemmodellering vil således være at starte med modellering af den planlagte systemudviklingsproces omfattende systemudviklingsaktiviteter og systemudviklingsprodukter i form af forskellige generelle delsystemmodeller eller forskellige versioner af en generel systemmodel.

Systemudviklingsaktiviteterne og systemudviklingsprodukterne kan modelleres som ikke nærmere definerede, overordnede delsystemer indenfor hvilke den efterfølgende generelle systemmodellering foregår. Alternativt kan systemudviklingsaktiviteterne og systemudviklingsprodukterne modelleres mere detaljeret med definition af deres struktur og indhold - for eksempel i forhold til systemudviklingsorganisationens standarder for systemudviklingsforløb eller standard systemoudviklingsmetodologier.



Detaljeret systemudviklingsproces modellering er kompleks med en systemmodel omfattende en række forskellige systemelementer i form af aktører, roller, artifakter og processer; en række forskellige mål i form af forståelse og kommunikation samt proces styring, forbedring, vejledning og automatisering; en række forskellige perspektiver i form af for eksempel funktioner, adfærd, organisation og information.

Med generel systemmodellering er der stor fleksibilitet til at omfatte alle disse aspekter med multidimensionel, hierarkisk systemmodellering og det generelle systemmodelleringssprog muliggør systemmodellering af såvel uformelle, sociale processer og relationer som formelle krav, processer og relationer.

Organisation modellering

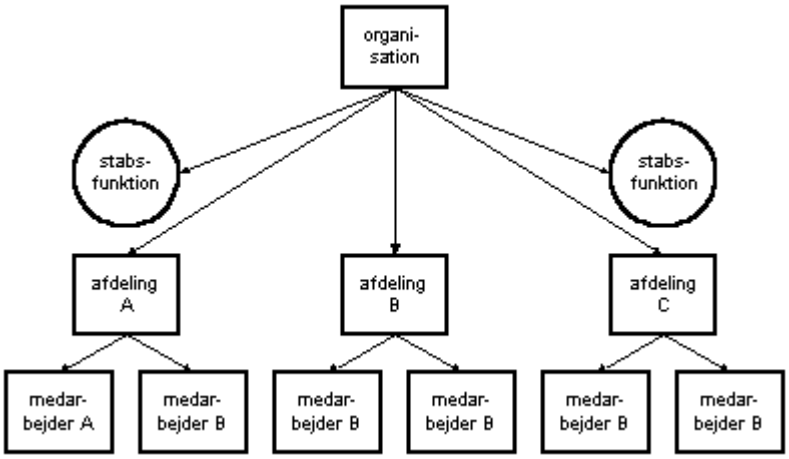
prev next up

Mintzberg [1983] Structure in Fives

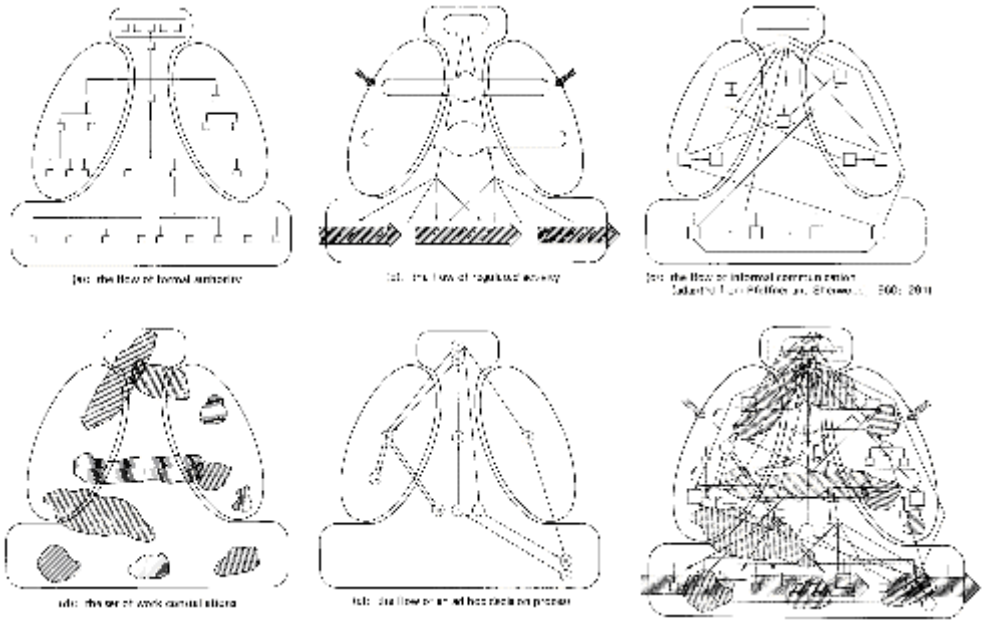
Wellman & Berkowitz [1988] Social Structures: A Network Approach

I forbindelse med systemudvikling af administrative systemer har systemudviklerne typisk ikke detaljeret kendskab til kundens organisation som det administrative system skal fungere i. For at få en vis forståelse af systemets bredere kontekst udover de umiddelbare omgivelser til systemet kan en overordnet beskrivelse af organisationen for eksempel i form af en simpel, eksisterende organisationsmodel være hensigtsmæssig. For større administrative systemer kan detaljeret organisationsanalyse desuden være nødvendigt og en del af det samlede systemudviklingsforløb.

En simpel organisationsmodel kan let modelleres med generel systemmodellering i form af et whole/part hierarki af linie- og stabsfunktion delsystem typer for afdelinger og personer.

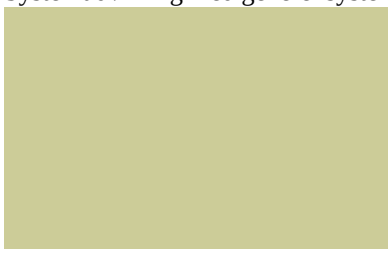


Detaljerede organisationsmodeller som Mintzbergs fem/seks-dimensionelle organisationsstrukturer kan ligeledes modelleres med generel systemmodellering i form af en multidimensionel generel systemmodel, hvor et simpelt organisationshierarki danner udgangspunktet med supplerende lag/dimensioner for forskellige typer af relationer og grupperinger som tilsammen udgør en kompleks organisationsmodel.



Kilde: Mintzberg [1983] Structure in Fives

De forskellige lag/dimensioner i organisationsmodellen udgør netværk af forskellige relationer mellem organisationens afdelinger og medarbejdere, som kan identificeres ved hjælp af eller danne udgangspunkt for mere detaljeret analyse af de indbyrdes relationer



mellem afdelingerne og medarbejderne igennem social strukturel analyse baseret på matematiske grafer. En sådan analyse for eksempel identificere interessante grupperinger af medarbejdere, stærke/svage koblinger mellem medarbejdere og afdelinger samt nøglemedarbejdere og -afdelinger af og andre forhold der kan have central betydning for implementeringen og anvendelsen af systemet der skal udvikles.

Problemsituation modellering

prev next up

Checkland [1990] Soft Systems Methodology in Action

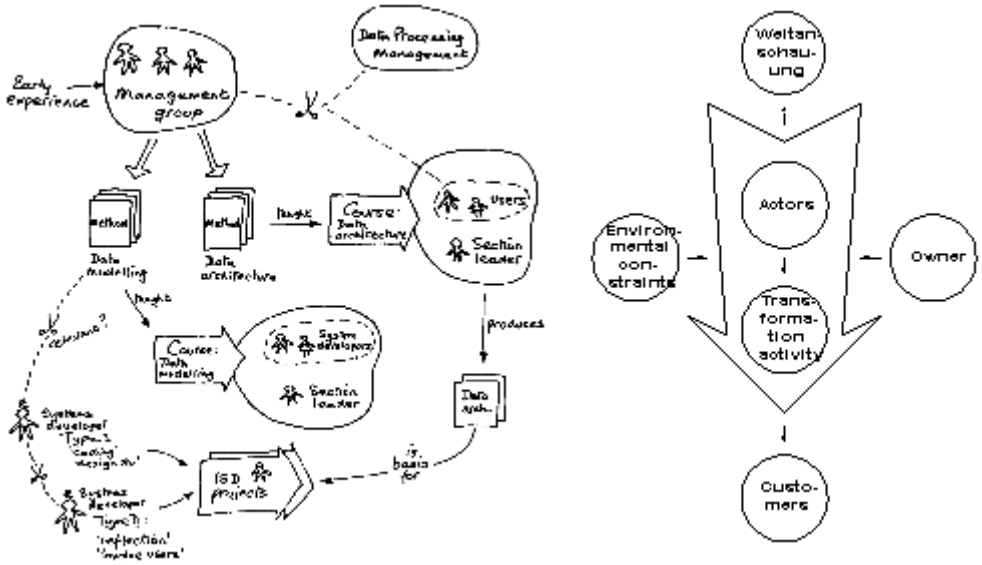
Wilson [1990] Systems: Concepts, Methodologies and Applications

Problemsituationen der har givet anledning til et systemudviklingsprojekt er typisk ikke detaljeret afklaret, og det er således hensigtsmæssigt/nødvendigt at systemudviklingsforløbet omfatter en analyse af problemsituationen. En sådan analyse kan passende udarbejdes ved hjælp af "rich picture" og "relevante systemer" modellering.

"Rich picture" modelleringen kan ske med udgangspunkt i en simpel eller detaljeret organisationsmodel, hvor et udsnit af relevante medarbejdere/afdelinger og relationer mellem disse kan udgøre den indledende "rich picture" model. Dermed vil problemsituationen være indplaceret i en større organisatorisk sammenhæng og igennem den videre "rich picture" modellering vil ændringer/udvidelser umiddelbart kunne sammenholdes/koordineres med modellen af problemsituationens omgivelser.

En "rich picture" model består af nogle grundlæggende elementer men kan i princippet indeholde hvad som helst af elementer og relationer der findes relevante. Generel systemmodellering er således velegnet til "rich picture" modellering ved umiddelbart at kunne omfatte hvad som helst men samtidigt give mulighed for en vis formalisering igennem anvendelse af delsystem typer/subtyper.

En problemsituation kan omfatte mange forskellige og eventuelt uforenelige synsvinkler uden nogen umiddelbart "rigtige" og for valg imellem disse kan en præcisering af synsvinklerne i form af "relevante systemer" bestående af en CATWOE formalisering og en dataflowdiagram lignende konceptuel model være hensigtsmæssig.



Kilde: Checkland [1990] Soft Systems Methodology in Action

Med generel systemmodellering kan forskellige "relevante systemer" modelleres som multidimensionelle modeller ovenpå organisation og problemsituation modellerne. Igennem det efterfølgende systemudviklingsforløb med udgangspunkt i et eller flere udvalgte "relevante systemer" vil systemet dermed løbende kunne sammenholdes med såvel organisation, problemsituation, de udvalgte "relevante systemer" og de fravalgte "relevante systemer" med henblik på bedre forståelse af systemets formål og omgivelser og afklaring af spørgsmål.

"Rich picture" modellen og de udvalgte "relevante systemer" modeller udgør et godt udgangspunkt for formuleringen af formålene med og kravene til systemet der skal udvikles i en detaljeret kravspecifikation. Disse modeller udgør ligeledes et godt udgangspunkt for en efterfølgende systemanalyse.

Kravspecifikation modellering

prev next up

Jackson [1995] Software Requirements & Specifications

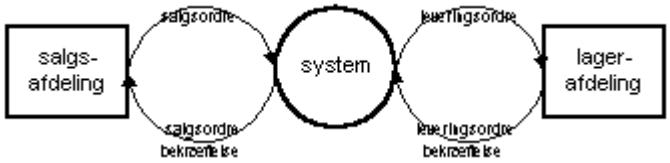
Curtis, Kellner & Over [1992] Process Modeling

Mylopoulos, Chung & Yu [1999] From Object-Oriented to Goal-Oriented Requirements Analysis

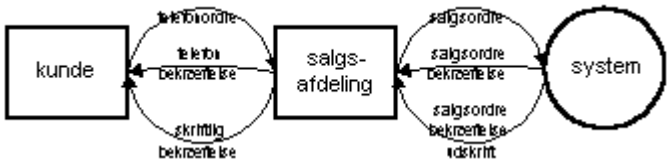
Kravspecifikationen bør beskrive kravene til systemet som krav til relationerne mellem systemet set som en blackbox og systemets omgivelser. Dermed kan kravene og løsningen holdes separerede men alligevel relaterede.

"Rich picture" modellen og de udvalgte "relevante systemer" modeller udgør et godt udgangspunkt for formuleringen af formålene med og kravene til systemet der skal udvikles. Disse kan med whole-part hierarkier kombineres og tilpasses til en kontekst model for systemet og/eller et antal brugsscenarier omfattende relevante interessenter og relationer til systemet.

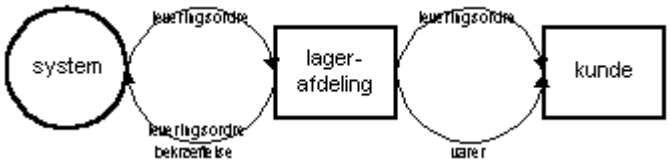
Kontekstdiagram



Ordre brugsscenarie



Levering brugsscenarie



De detaljerede krav kan modelleres igennem nedbrydning af modellernes interessenter og relationer til enkeltkrav. Systemets inddata og uddata kan nedbrydes til veldefinerede, detaljerede delsystemer i form af computerbaserede datatyper suppleret med verbale beskrivelser og kvantitative mål for andre krav som kapacitet og svartider. Relationer mellem systemet samt de primære og sekundære interessenter kan detaljeres med formel eller uformel proces modellering Alle krav bør kunne formuleres som krav i forbindelse med interessenter og interessenters relationer til systemet.

Med generel systemmodellering kan kravspecifikationen således integreres med problemsituation modelleringen for umiddelbar indplacering i en større kontekst. Kravspecifikationen kan ligeledes formuleres som en struktureret model, der kan udgøre udgangspunktet for efterfølgende systemanalyse, systemdesign og implementering, selvom kravspecifikationen typisk i vid udstrækning vil bestå af verbale beskrivelser.

Systemanalyse modellering

prev next up

Avison & Fitzgerald [1995] Information Systems Development

Yourdon [1989] Modern Structured Analysis

Champeaux, Lea & Faure [1993] Object-Oriented System Development

Systemanalysen omfatter modellering af relevante aspekter og elementer indenfor systemet; i interaktionen mellem systemet og omgivelserne; samt i omgivelserne for den nødvendige, detaljerede forståelse af systemet.

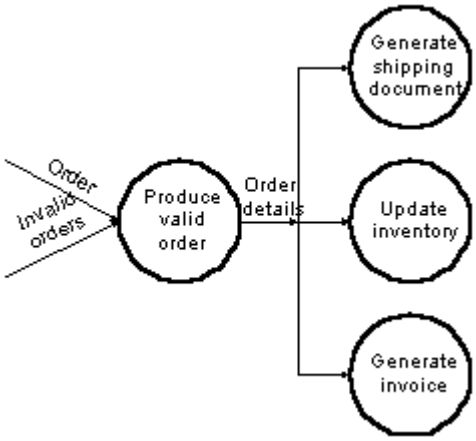
Med problemsituation modellering i form af "rich picture" og "relevante systemer" modeller samt kravspecifikation modelleringen er der et godt udgangspunkt for yderligere nedbrydning og detaljering i forbindelse med systemanalyse modelleringen. Systemanalysen omfatter typisk modellering af "logiske" data og processer samt eventuelt aktører, fysiske artifakter og fysiske processer.

Med generel systemmodellering kan systemanalyse modellering laves med hierarkisk modellering af data og processer samt relationer mellem disse. Yderligere aspekter som aktører, fysiske artifakter og fysiske processer kan ligeledes medtages i én multidimensionel generel systemmodel.

De forskellige traditionelle systemudviklingsmetodologier er typisk bundet til ét systemmodelleringsparadigme i form af funktions-, data- eller objektorienteret systemmodellering, men med generel systemmodellering kan disse blot betragtes som forskellige strukturerings- og præsentationsformer af de samme grundlæggende elementer.

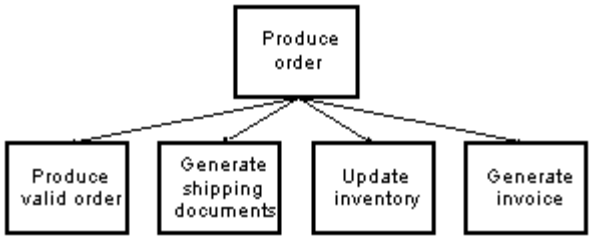
Generel systemmodellerings whole-part og type-instance strukturer samt relationer mellem delsystemer dækker umiddelbart mange traditionelt anvendte systemanalyse modeller som dataflow diagrammer og hierarkiske proces og data diagrammer.

Struktureret systemudvikling - dataflow diagram

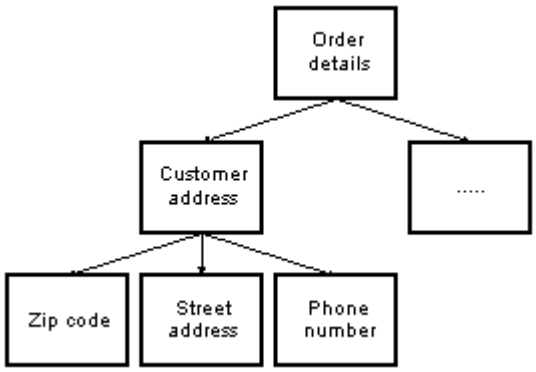


Kilde: Yourdon [1989] Modern Structured Analysis

Funktionsorienteret proceshierarki diagram

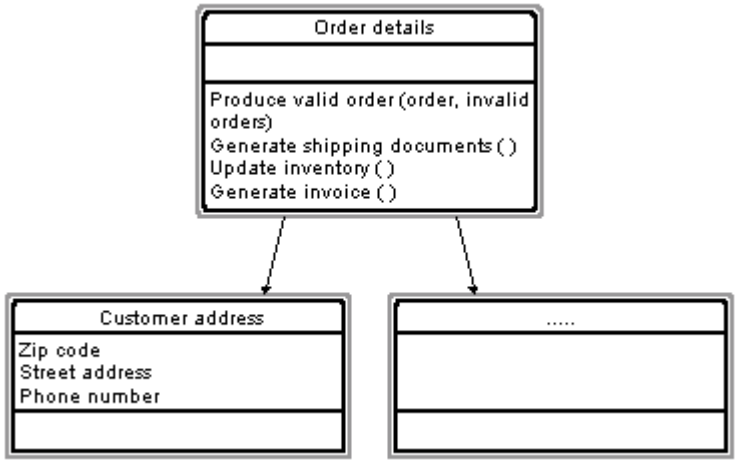


Dataorienteret datahierarki diagram



Andre traditionelt anvendte systemmodelleringsstrukturer som entiteter-relationer samt objekter kan modelleres eksplicit med generel systemmodellering, men sådanne strukturer kan også umiddelbart udledes af andre generelle systemmodeller. Entitet-relation strukturer kan udledes af multidimensionelle data hierarki modellet, og ved systemmodellering af data og processer med henholdsvis data og proces delsystem type-instance relationer kan processer grupperes med direkte input-output relaterede data som objekter.

Objektorienteret systemudvikling - objekthierarki diagram



Generel systemmodellering giver således fleksible muligheder for systemanalyse modellering samt strukturering og præsentation for fremhævning af forskellige aspekter ved systemanalysen. Systemanalyse modelleringen kan desuden integreres med problemsituation og kravspecifikation modelleringen samt danne udgangspunkt for efterfølgende systemdesign modellering.

Systemdesign modellering

prev next up

Butler, Esposito & Hebron [1999] Connecting the Design of Software to the Design of Work

Yourdon [1989] Modern Structured Analysis

Champeaux, Lea & Faure [1993] Object-Oriented System Development

Systemdesign er orienteret mod systemmodellering af en løsning der kan implementeres som et computerbaseret system samt omkringliggende processer igennem formulering af algoritmer og manuelle procedurer samt detaljering af formkrav og fejlhåndtering.

Systemdesignet tager udgangspunkt i systemanalysens systemmodel med valg af hvilke dele af systemet der kan og skal implementeres som et computerbaseret system, samt detaljering og præcisering af hvordan denne implementering kan ske i form af et computerbaseret system og omgivelsernes interaktion med det computerbaserede system.

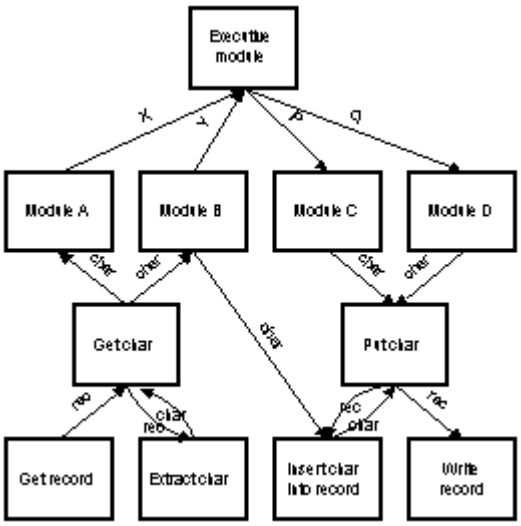
Systemets data detaljeres og præciseres med computerorienterede datatyper og datastrukturer, og systemets processer detaljeres og præciseres med computerorienterede instruktioner til bearbejdning af data og styring af processerne.

De forskellige traditionelle systemudviklingsmetoder er som ved systemanalyse også typisk bundet til ét systemmodelleringsparadigme i form af funktions-, data- eller objektorienteret systemmodellering men med generel systemmodellering kan disse blot betragtes som forskellige strukturerings- og præsentationsformer af de samme grundlæggende elementer.

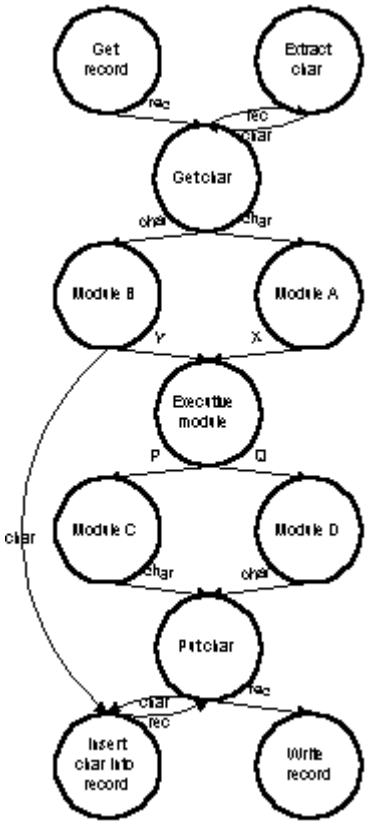
Systemdesign datamodellering ved traditionel systemudvikling omfatter typisk entitet-relation modellering og normalisering af data med henblik på implementering af data i en relationsdatabase. Med generel systemmodellering kan datamodellering også eksplicit laves som entitet-relation modeller, men datamodellering kan/bør i stedet for laves med de generelle whole-part og type-instance relationer, som kan transformeres til en entitet-relation model, hvis data skal implementeres i en database.

Ved traditionel systemudvikling omfatter systemdesign procesmodellering typisk en form for hierarkisk processtrukturdiagram med overførsel af data mellem styrende og bearbejdende delprocesser. Med generel systemmodellering kan sådan et proces hierarki umiddelbart modelleres med whole-part relation suppleret med system-relation relationer for de overførte data.

Overførsel af data mellem hierarkiske niveauer er dog ikke "naturlig" med generel systemmodellering, da det indikerer et "fysisk" hierarki med imperative programmeringssprog orienterede funktioner/procedurer der kalder andre funktioner/procedurer. Ved at se bort fra de hierarkiske whole-part relationer kan den hierarkiske processtruktur transformeres til et tilsvarende, mere generelt netværk af delsystemer. Disse kan eventuelt opdeles i et hierarki af hensyn til abstraktion og gruppering af tæt relaterede delsystemer, men dette er ikke ensbetydende med at de skal implementeres som hierarkiske programmeringskomponenter i form af procedurer/funktioner.

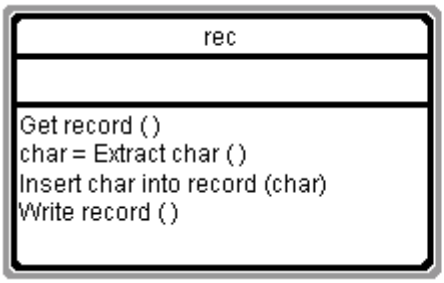


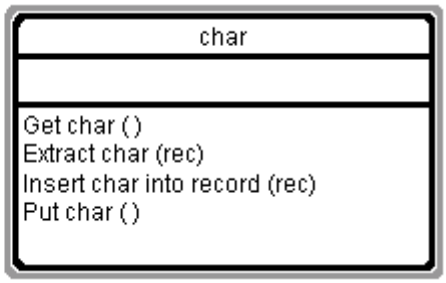
Kilde: Yourdon [1989] Modern Structured Analysis



Systemdesignet bør ikke være bundet til en bestemt type programmeringssprogs programmeringsmodel. Beslutninger om sådanne implementeringsdetaljer bør træffes i forbindelse med fastlæggelse af implementeringen eller systemarkitekturen med identifikation af relevante programmer, moduler og procedurer/funktioner (hvis der vælges en imperativ implementering).

På tilsvarende måde som ved systemanalyse kan systemmodellering af data og processer desuden transformeres og præsenteres objektorienteret med gruppering af processer omkring direkte input-output relaterede data.





Med generel systemmodellering kan systemdesign således ske som en integreret detaljering af systemanalysen og uafhængigt af eventuelt senere valg af systemarkitektur og implementering.

Systemarkitektur modellering

prev next up

Medvidovic & Taylor [1998] Separating Fact from Fiction in Software Architecture

Brown, Malveau, McCormick & Mowbray [1998] Anti Patterns

Davis & Williams [1997] Software Architecture Characterization

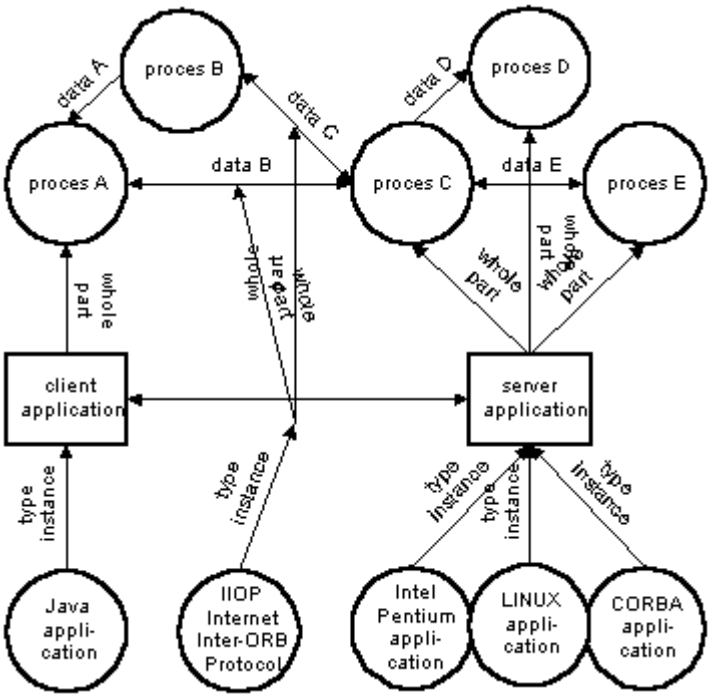
Abowd, Allen & Garlan [1993] Using Style to Understand Descriptions of Software Architecture

Métayer [1996] Software Architecture Styles as Graph Grammars

Lung [1998] Software Architecture Recovery and Restructuring through Clustering Techniques

Med systemarkitektur modellering opdeles systemet i et antal overordnede delsystemer i form af selvstændige, kommunikerende computere, programmer, processer eller komponenter, og det fastlægges hvordan kommunkationen mellem delsystemerne skal foregå.

Systemarkitektur modellering kan betragtes som en form for overordnet systemdesign med fokus på den fysiske gruppering og implementering af delsystemerne, og deres indbyrdes kommunikationsmedier og protokoller. Systemmodelleringen af dette kan ske ved at gruppere de enkelte delsystemer med whole-part relationer samt at definere formen af disse med type-instance relationer.



Systemarkitekturen kan defineres specifikt for det enkelte system eller med udgangspunkt i standard system- og softwarearkitekturer i form af mønstre der er formuleret som generelle systemmodeller.

En systemarkitektur omfatter en række forskellige aspekter som for eksempel domæne, komponenter, platforme, grænseflader og kontekst med forskellige relevans for for eksempel systemets brugere, systemudviklere og driftansvarlige. Med systemarkitekturen modelleret som én integreret generel systemmodel kan de forskellige aspekter filtreres og præsenteres til forskellige anvendelsesformål og brugere/systemudviklere.

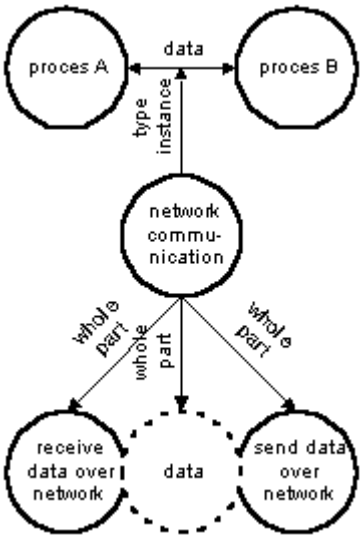
Ved komplekse systemer og systemarkitekturer kan generelle systemmodeller desuden analyseres med matematisk graf baserede metoder til strukturering og omstrukturering af systemer til hensigtsmæssige system- og softwarearkitekturer.

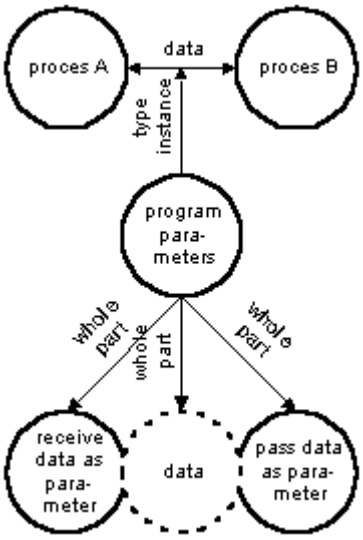
Systemarkitekturen definerer rammerne for implementeringen af de enkelte delsystemer ved at disse i sidste ende skal nedbrydes til basale delsystemer for en bestemt type af computer processor, styresystem, programarkitektur og kommunikationsarkitektur. Systemarkitekturen er således overordnet systemdesignet men også tæt forbundet med den detaljerede implementering. Systemdesignet bør dog såvidt muligt være uafhængigt af såvel systemarkitekturen som implementering med fokus på generel design af løsningen, og systemdesignet bør ikke nødvendigvis skulle ændres ved ændring af systemarkitekturen.

Dette kan opnås ved at anvende en standard samling af basale delsystemer svarende til for eksempel anvendelse af POSIX styresystem systemkald og C programmeringssprog funktionsbibliotek funktioner, der er implementeret med den samme eksterne grænseflade på mange forskellige systemplatforme. I det omfang at implementering sker med ikke-standardiserede basale delsystemer må implementeringen dog ændres til den ændrede systemarkitektur, eller de manglende basale delsystemer i systemarkitekturen (i forhold til den tidligere systemarkitektur) må implementeres.

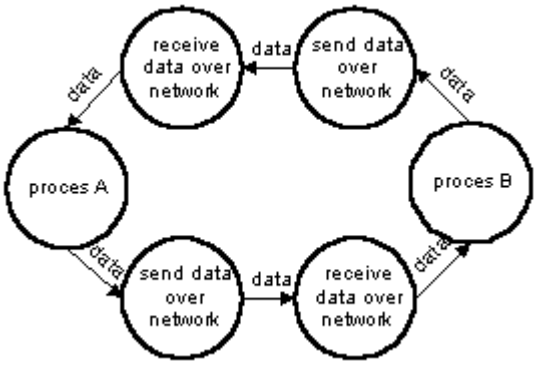
Hver enkelt type af systemarkitektur kan desuden være specificeret og implementeret med de specielle formater og funktioner, der er nødvendige i forbindelse med den enkelte systemarkitektur udover den generelle programkode, som dermed eventuelt automatisk kan indpakkes/parres med de systemarkitektur specifikke formater og funktioner. En dataflow relation i systemdesignet og implementeringen kan for eksemple således være transparent i forhold til systemarkitekturen ved at en netværkkommunikation systemarkitektur type omfatter standard sende og modtage funktioner svarende til en program systemarkitektur type der omfatter standard funktioner for overførsel og modtagelse af programparametre.

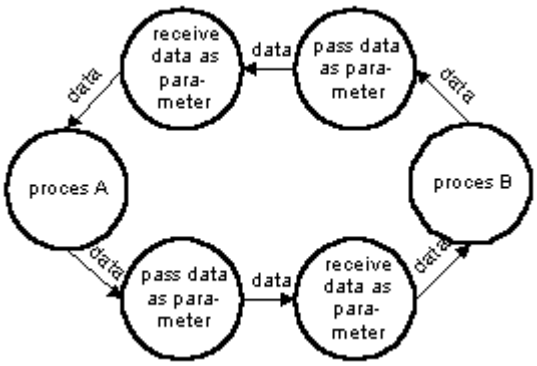
Systemarkitektur systemmodel





Aggregeret systemmodel





Med generel systemmodellering kan systemarkitektur således modelleres som supplerende dimension(er) på systemmodellen for systemdesign og implementering, og indenfor visse rammer kan generel systemdesign og systemimplementering ske uafhængigt af men integreret med systemarkitektur modellering med forskellige systemarkitekturer for stor fleksibilitet ved valg og ændring af systemarkitektur for et system.

Højniveau program modellering

prev next up

Finkel [1995] Advanced Programming Language Design

Appel [1998] Modern Compiler Implementation in Java

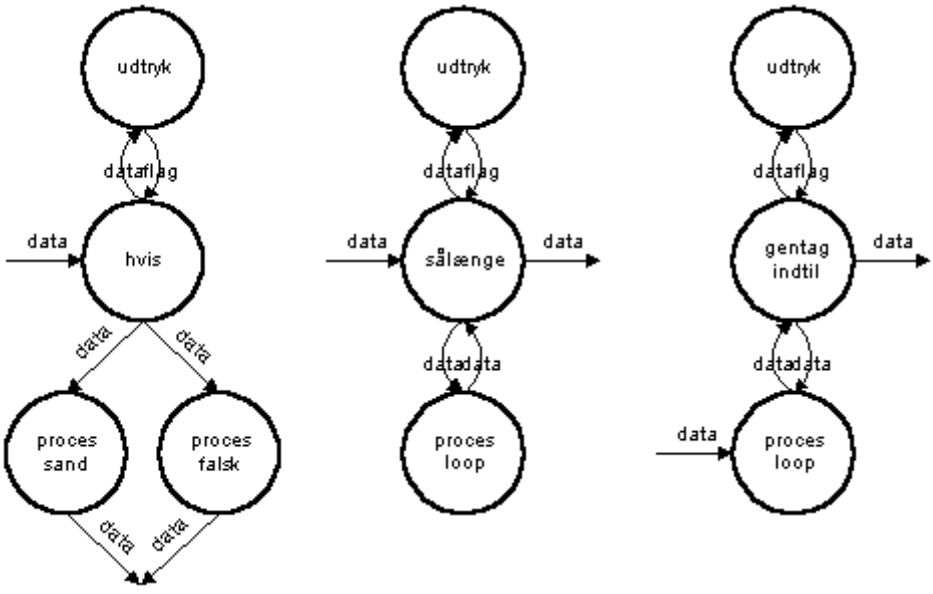
Morrison [1994] Flow-Based Programming

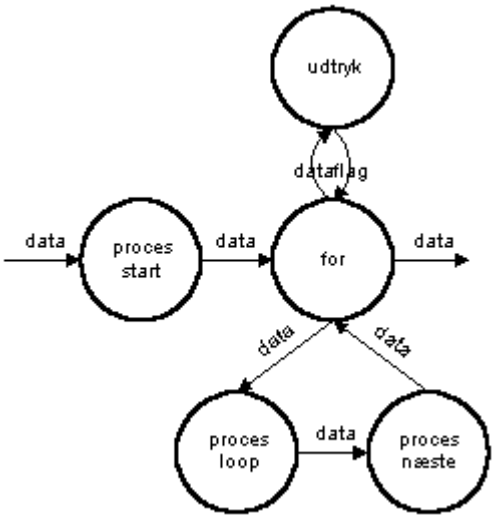
Højniveau programmering detaljerer systemdesignet til sekvenser af basale data og kommandoer i forhold til et givent programmeringssprog ved traditionel systemudvikling. Sådanne basale data og kommandoer omfatter typisk et antal basale datatyper som for eksempel heltal, decimaltal, tekst og sammensatte datastrukturer, samt kommandoer til at erklære forekomster af disse datatyper i form af variable. Disse variable anvendes i forbindelse med basale kommandoer til bearbejdning i form af for eksempel kopiering, addition og subtraktion. Derudover omfatter programmeringssprog kontrol kommandoer til styring af programeksekveringen med iteration og selektion samt kommandoer til gruppering og abstraktion i form af procedurer og funktioner. Endeligt omfatter programmeringssprog eksplicitte eller implicitte kommandoer til allokering og styring af systemressourcer i form af for eksempel lager og filer samt ind- og uddata enheder.

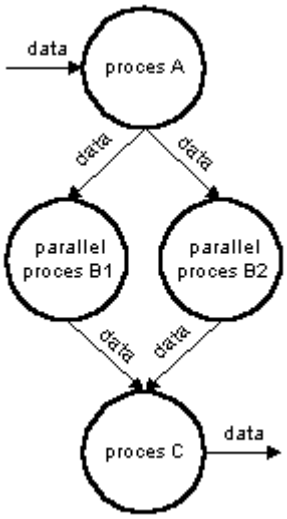
Der findes nærmest utallige forskellige programmeringssprog indenfor et antal forskellige programmeringsparadigmer med varierende syntaks for og varierende fokus på forskellige datatyper samt bearbejdnings- og kontrolkommandoer. De er dog alle abstraktioner for effektiv anvendelse af ofte benyttede programkonstruktioner i forbindelse med forskellige programtyper baseret på de samme basale, lavniveau computerinstruktioner, der benyttes ved fortolkning eller kompilering af programkoden.

Traditionelle programmeringssprogs programkonstruktioner består typisk i blokke og sekvenser af enkelte programinstruktioner men disse dækker dog over mere komplekse netværk af programinstruktioner idet de enkelte programinstruktioner udover de simple, sekventielle relationer er relaterede via de data de anvender og bearbejder. Sådanne komplekse netværk udledes for eksempel i forbindelse med optimering af programkonstruktioner ved kompilering af højniveau programkode til lavniveau programkode.

Ved generel systemmodellering kan programkonstruktioner tilsvarende modelleres med netværk af delsystemer svarende til de enkelte programinstruktioner samt hierarkier af disse. En sådan netværk modellering af programkonstruktioner afspejler bedre de reelle bindinger mellem de enkelte programinstruktioner samt programinstruktioner uden indbyrdes bindinger som derfor kan ekskveres parallelt.







Med generel systemmodellering, der omfatter systemmodellering med hierarkier og netværk af lavniveau computerinstruktioner samt generelle abstraktionsmekanismer, er det således også muligt at definere delsystemer svarende til de basale data og kommandoer i højniveau programmeringssprog. Ved generel systemmodellering behøver højniveau program modellering dermed ikke at være bundet til et enkelt programmeringssprogs basale koncepter og syntaks. Der kan umiddelbart defineres og anvendes en vilkårlig sammensætning af forskellige basale programkonstruktioner. Sådanne basale programkonstruktioner bør være defineret som standard delsystemmodeller der umiddelbart kan anvendes i forbindelse med implementering af computerbaserede systemer.

Gode programmeringssprog er dog mere end blot en samling tilfældige, basale programkonstruktioner men snarere en omhyggeligt sammensat samling af basale programkonstruktioner orienteret mod implementering af bestemte programtyper selvom mange programmeringssprog kan betegnes som generelle programmeringssprog. Det er dermed eventuelt ikke hensigtsmæssigt at blande for mange og for forskellige basale koncepter ved generel systemmodellering. Ved generel systemmodellering har man dog altid muligheden for at anvende eller definere andre basale programkonstruktioner, hvis det er hensigtsmæssigt eller nødvendigt.

Generel systemmodellering af programkonstruktioner er her præsenteret som dataflow lignende diagrammer, men dette er dog primært for at fremhæve de matematisk graf orienterede strukturer, og det er ikke ensbetydende med at generel systemmodellering af programkonstruktioner er bundet til dataflow orienteret visuel programmering. Generel systemmodellering er ikke bundet til en bestemt syntaks til repræsentation af programmer, idet alle programkonstruktioner er defineret som generelle systemmodeller uden bunden syntaks.

Der kan dermed defineres forskellige præsentationsformer for generelle systemmodeller for programkonstruktioner, der præsenterer programkonstruktionerne i et vilkårligt format som for eksempel i form af dataflow lignende diagrammer eller programkildetekst svarende til syntaksen for et traditionelt programmeringssprog. Ved præsentation i programkildetekst format kan generelle systemmodeller umiddelbart udveksles med traditionelle, specialiserede systemudviklingsværktøjer, og systemudviklere kan programmere ved hjælp af velkendte, effektive programmeringssprog. Dette forudsætter dog at den generelle systemmodel kun omfatter programkonstruktioner, det umiddelbart er muligt at udtrykke med det pågældende programmeringssprog eller at der er defineret specielle delsystemer i form af funktioner/procedurer for de manglende programkonstruktioner.

Med generel systemmodellering kan højniveau program modellering således ske som detaljering af systemdesign i én integreret, generel systemmodel med de samme generelle systemkoncepter.

Lavniveau program modellering

prev up down

Lavniveau program modellering eksempel

Tanenbaum [1990] Structured Computer Organization

Motorola [1986] M68000 8-/16-/32-Bit Microprocessors Reference Manual

Lavniveau program modellering er detaljering af højniveau programkonstruktioner til af computeren direkte eksekverbare programinstruktioner i form af sekvenser af bytekoder til repræsentation af disse programinstruktioner samt data og adresser for data i computerens lager.

Lavniveau program modellering svarer til højniveau program modellering blot med simplere programkonstruktioner og datatyper der er bundet til en bestemt computertypes processor og anden hardware. Datatyperne omfatter simple data som bits og bytes, og al bearbejdning af data skal ske med et givent antal variable i form dataregistre samt adresserbart lager i computeren, Programinstruktionerne omfatter tilsvarende simple programinstruktioner til kopiering og sammenligning samt matematisk og logisk bearbejdning af disse data.

Dermed kan lavniveau program modellering ske med generel systemmodellering på tilsvarende måde som højniveau program modellering med netværk af programinstruktioner der er relateret via de data de anvender og bearbejder. De enkelte programinstruktioner og data er dog begrænset til et givent sæt af delsystemer der er defineret for en bestemt computertype. Disse delsystemer skal være nedbrudt til aksiomatiske delsystemer i form af for eksempel bits, som et generelt systemmodelleringsværktøj kan anvende til at generere direkte eksekverbare computer programfiler. Disse basale computerspecifikke delsystemer bør være defineret som standard delsystemmodeller for forskellige computertyper der umiddelbart kan anvendes i forbindelse med implementering af computerbaserede systemer.

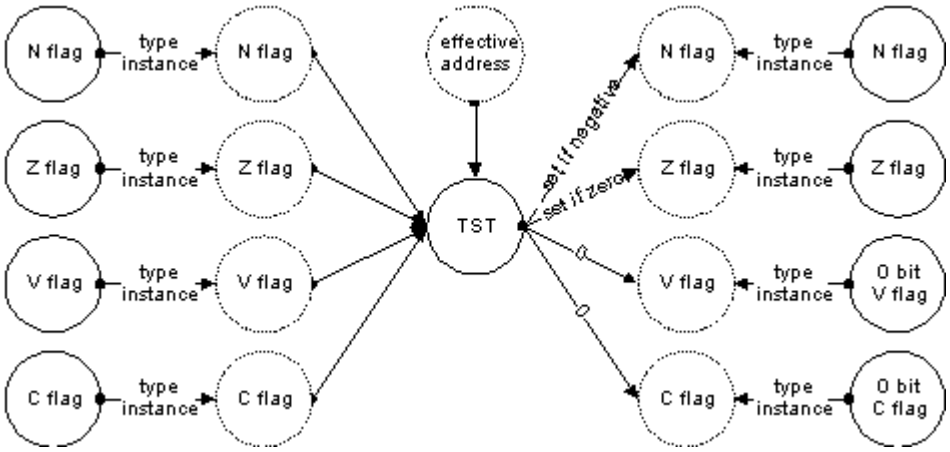
Som for højniveau program modellering er lavniveau program modellering heller ikke bundet til bestemte præsentationsformer. Dataflow lignende matematisk graf baserede præsentationer fremhæver de reelle relationer mellem de enkelte programinstruktioner, mens at tekstuel præsentation i programkildetekst format er mere kompakt og umiddelbart muliggør udveksling af generelle systemmodeller med traditionelle, specialiserede systemudviklingsværktøjer.

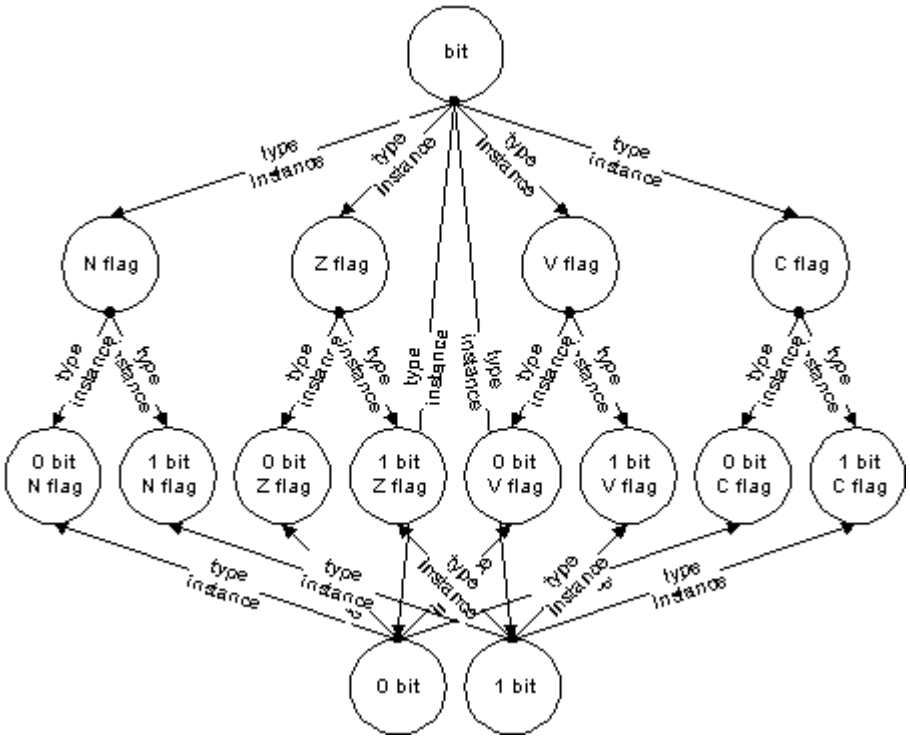
Præsentation af lavniveau programmer (og højniveau programmer) som netværk af programinstruktioner viser at selv tilsyneladende forholdsvis simple programmer eventuelt omfatter mange relationer imellem de enkelte programinstruktioner i et komplekst netværk som det ses af et simpelt eksempel med generel systemmodellering af en funktion til kopiering af en blok data fra en adresse til en anden i computerens lager.

Den matematisk graf baserede præsentation af den generelle systemmodel bliver således forholdsvis omfattende og kompleks, og generel systemmodellering af lavniveau programmer som matematisk graf baseret modellering med generelle systemmodelleringsværktøjer er dermed eventuelt ikke hensigtsmæssig. Omvendt viser den matematisk graf baserede repræsentation den reelle kompleksitet, der ellers skal udledes og håndteres mentalt af systemudviklerne.

De enkelt basale programinstruktioner for en computerprocessor kan defineres meget detaljeret og præcist med generelle type-instance relationer for ind- og uddata. Dette svarer til formel specifikation af præ og post betingelser og sådanne definitioner kan således også danne udgangspunkt for formel specifikation og verifikation af (dele af) det samlede system. For de overliggende niveauer af systemmodellen vil præ og post betingelser således automatisk kunne beregnes igennem aggregering af de basale programinstruktioners præ og post betingelser, og dermed kan systemmodellen analyseres for konsistens.

Detaljeret definition af Motorola 68000 processor TST maskinkode programinstruktionen. (Test om værdien af en lagercelle er nul, positiv eller negativ og angiv resultatet i processorens dataregister flag.)





Det er således også muligt at repræsentere lavniveau program modellering som generelle systemmodeller ved hjælp af de generelle koncepter i én integreret systemmodel for alle systemudviklingsaktiviteter.

Konklusion på systemudvikling med generel systemmodellering

prev up level

Lagring af generelle systemmodeller

Generelle præsentationsværktøjer

Generelle analyseværktøjer

Generelle vs. specialiserede værktøjer

Generel systemmodellering systemudviklingsproces

Systemudvikling af computerbaserede systemer omfatter store og komplekse systemmodeller som ikke selv er praktisk mulige uden computerunderstøttede systemmodelleringsværktøjer, og understøttelse af det formulerede, generelle systemmodelleringssprog med generelle systemmodelleringsværktøjer til lagring, præsentation, analyse og bearbejdning af generelle systemmodeller er således nødvendige.

Standarder for lagring af og grænseflader for bearbejdning af generelle systemmodeller er nødvendige for at muliggøre udveksling og deling af systemmodeller mellem systemudviklere og systemmodelleringsværktøjer. Generelle systemmodeller stiller i sig selv ikke specielle krav til lagring og grænseflader, og standarder kan således vilkårligt fastlægges med udgangspunkt i forskellige, eksisterende standarder.

Selvom generelle systemmodeller udelukkende består af simple, matematisk graf baserede netværk og hierarkier kan disse repræsentere meget forskellige og komplekse strukturer som også kan præsenteres på meget forskellige måder fra verbale og matematisk/logiske beskrivelser til to- og tredimensionelle visualiseringer. De forskellige, generelle præsentationsformer for generelle systemmodeller fremhæver forskellige aspekter ved de generelle systemmodeller, som hver især og i kombination er relevante for forskellige systemudviklingsaktiviteter og systemudviklere.

De matematisk graf baserede, generelle systemmodeller med en formelt defineret semantik giver desuden mulighed for helt eller delvist automatiseret analyse og bearbejdning af de generelle systemmodeller specielt med henblik på verifikation og forbedring af selve systemmodellerne, således at systemudviklerne i højere grad kan fokusere på de kvalitative og indholdsmæssige aspekter ved systemerne og deres indplacering i omgivelserne.

Da generel systemmodellering principielt skal kunne anvendes til modellering af vilkårlige systemer, kan det også anvendes til systemmodellering i forbindelse med specielle systemmodelleringsværktøjer således at generelle systemmodeller kan omfatte og integreres/kombineres med traditionelle, eksisterende systemmodelleringsværktøjer og systemmodeller. Dermed er det muligt gradvist at inddrage generel systemmodellering i systemudviklingsforløb samt fortsat at anvende effektive, specialiserede systemmodelleringsværktøjer.

Det formulerede, generelle systemmodelleringssprogs få, simple, generelle systemmodelleringskoncepter er tilstrækkelige til systemmodellering svarende til traditionel systemmodellering. Rammer, begrænsninger, standarder og mønstre for forskellige delsystemmodeller kan defineres som typer der udfyldes og detaljeres i forbindelse med konkrete systemmodeller. Genanvendelige delsystemer kan desuden modelleres som delsystemer, der kan anvendes som komponenter i andre systemer. Uformelle delsystemer kan modelleres som ikke videre definerede delsystemer uden klassificering og nedbrydning. Formelle delsystemer kan omvendt modelleres præcist med detaljeret klassificering og nedbrydning til veldefinerede, aksiomatiske delsystemer som for eksempel bits.

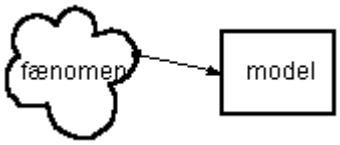
Generel systemmodellering er således meget fleksibel til systemudvikling af computerbaserede systemer med understøttelse af systemmodellering svarende til traditionel systemudvikling samt med udvidede muligheder for tilpasning af individuelle systemudviklingsforløb og integreret og forenklet systemmodellering.

Bouldings generelle systemtyper

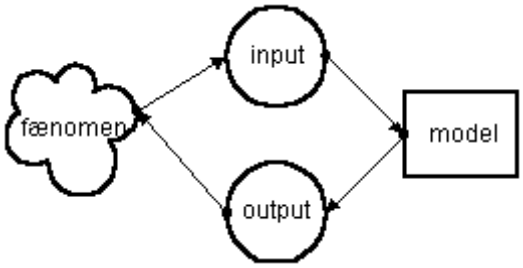
up

Boulding [1956] General Systems Theory - The Skeleton of Science

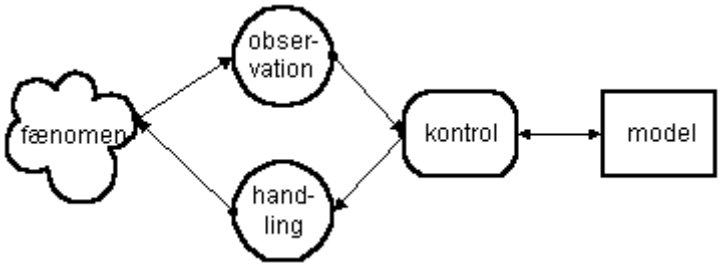
Statiske struktursystemer omfatter uforanderlige systemer med statiske relationer mellem elementerne.



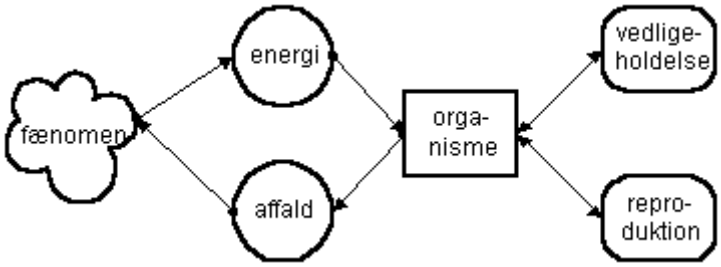
Simple dynamiske systemer omfatter systemer med deterministiske relationer mellem elementerne.



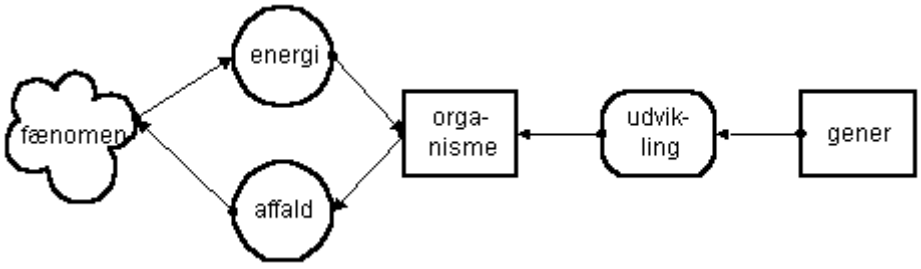
Kontrolsystemer omfatter systemer der regulerer sig selv udfra fra givne mål.



Selvorganiserende systemer omfatter systemer der vedligeholder sig selv igennem optagelse og udskillelse af materiale til/fra systemets omgivelser.

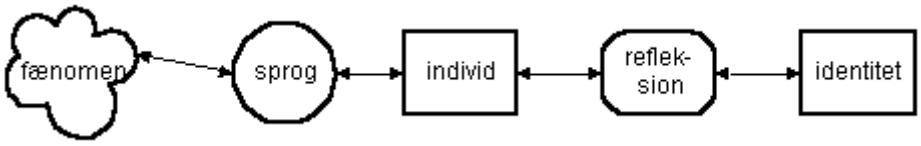


Genetisk-sociale systemer omfatter systemer bestående af specialiserede delsystemer og indbyggede gener, der bestemmer systemets udvikling.

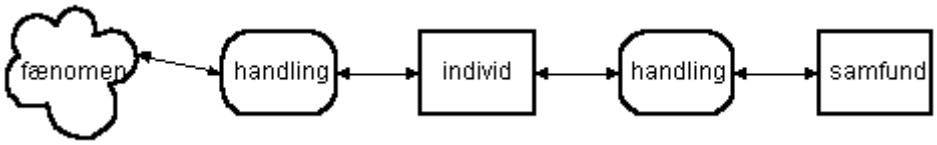


Selvorienterende (self-awareness) systemer omfatter systemer der orienterer sig og reagerer i forhold til systemets omgivelser.

Selvbevidste systemer omfatter systemer med selvrefleksion og symbolsk fortolkning/forståelse.



Sociale systemer omfatter systemer bestående af selvstændige delsystemer der påvirker hinanden igennem symbolsk kommunikation og handling.



Lavniveau program modellering eksempel

up down

push reg cli

clicks offset count

block copy more next

pop reg

Tanenbaum [1987] Operating Systems

Generel systemmodellering af en funktion til kopiering af en blok data fra en adresse til en anden i computerens lager.

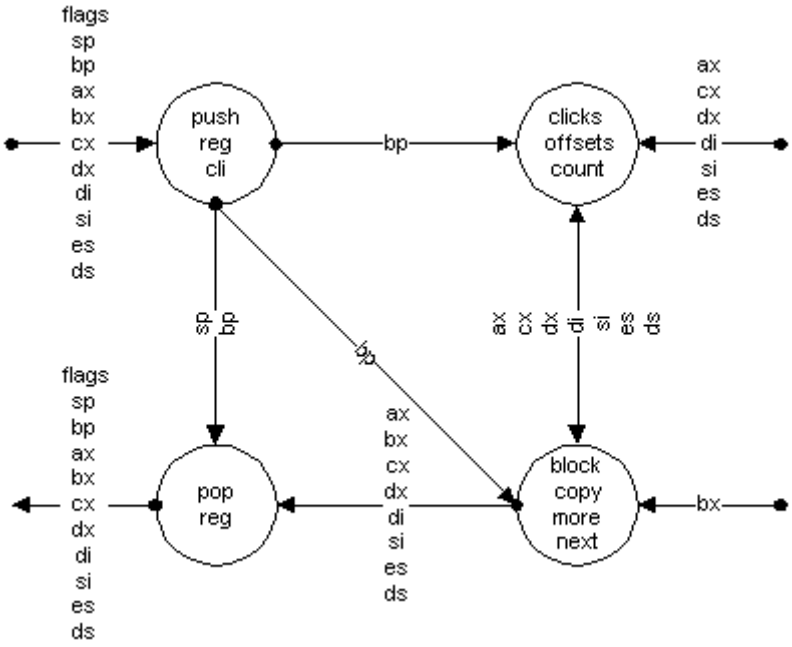
Maskinkode program:

```
| This routine copies a block of physical memory. It is called by:
|       phys_copy( (long) source, (long) destination, (long) bytecount)
_phys_copy:
        pushf           save flags
        cli             disable interrupts
        push bp         save the registers
        push ax         save ax
        push bx         save bx
        push cx         save cx
        push dx         save dx
        push si         save si
        push di         save di
        push ds         save ds
        push es         save es
        mov bp,sp       set bp to point to saved es

L0:     mov ax,28(bp)   ax = high-order word of 32-bit destination
        mov di,26(bp)   di = low-order word of 32-bit destination
        mov cx,*4       start extracting click number from dest
L1:     rcr ax,*1       click number is destination address / 16
        rcr di,*1       it is used in segment register for copy
        loop L1         4 bits of high-order word are used
        mov es,di       es = destination click

        mov ax,24(bp)   ax = high-order word of 32-bit source
        mov si,22(bp)   si = low-order word of 32-bit source
        mov cx,*4       start extracting click number from source
L2:     rcr ax,*1       click number is source address / 16
        rcr si,*1       it is used in segment register for copy
        loop L2         4 bits of high-order word are used
        mov ds,si       ds = source click

        mov di,26(bp)   di = low-order word of dest address
        and di,*0x000F  di = offset from paragraph # in es
        mov si,22(bp)   si = low-order word of source address
        and si,*0x000F  si = offset from paragraph # in ds

        mov dx,32(bp)   dx = high-order word of byte count
        mov cx,30(bp)   cx = low-order word of byte count

        test cx,#0x8000 if bytes >= 32768, only do 32768
        jnz L3          per iteration
        test dx,#0xFFFF check high-order 17 bits to see if bytes
        jnz L3          if bytes >= 32768 then go to L3
        jmp L4          if bytes < 32768 then go to L4
L3:     mov cx,#0x8000  0x8000 is unsigned 32768
L4:     mov ax,ax       save actual count used in ax; needed later

        test cx,*0x0001 should we copy a byte or a word at a time?
        jz L5           jump if even
        rep             copy 1 byte at a time
        movb            byte copy
        jmp L6          check for more bytes
```

```
L5:     shr cx,*1       word copy
        rep             copy 1 word at a time
        movw            word copy

L6:     mov dx,32(bp)   decr count, incr src & dst, iterate if needed
        mov cx,30(bp)   dx || cx is 32-bit byte count
        xor bx,bx       bx || ax is 32-bit actual count used
        sub cx,ax       compute bytes - actual count
        sbb dx,bx       dx || cx is # bytes not yet processed
        or cx,cx        see if it is 0
        jnz L7          if more bytes then go to L7
        or dx,dx        keep testing
        jnz L7          if loop done, fall through

        pop es          restore all the saved registers
        pop ds          restore ds
        pop di          restore di
        pop si          restore si
        pop dx          restore dx
        pop cx          restore cx
        pop bx          restore bx
        pop ax          restore ax
        pop bp          restore bp
        popf            restore flags
        ret             return to caller

L7:     mov 32(bp),dx   store decremented byte count back in mem
        mov 30(bp),cx   as a long
        add 26(bp),ax   increment destination
        adc 28(bp),bx   carry from low-order word
        add 22(bp),ax   increment source
        adc 24(bp),bx   carry from low-order word
        jmp L0          start next iteration
```

Kilde: Tanenbaum [1987] Operating Systems

Overordnet generel systemmodel

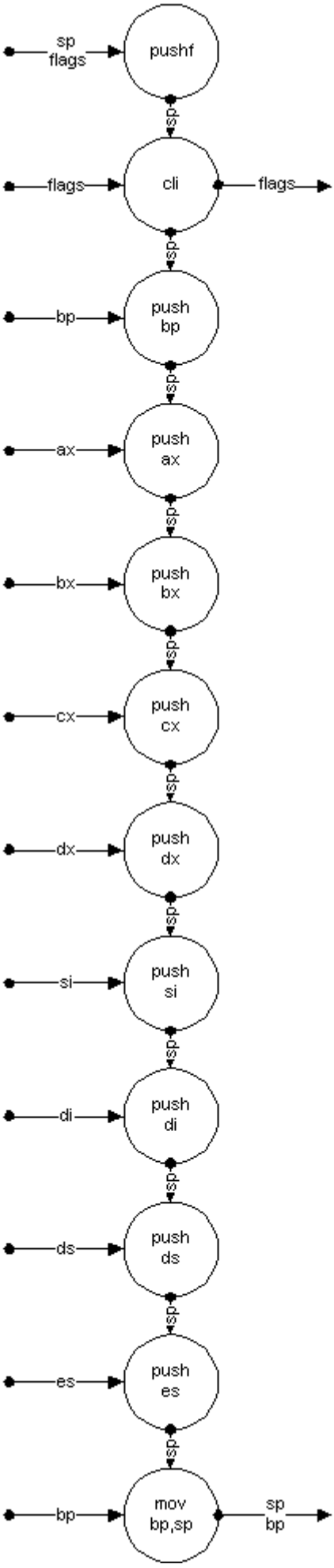


push reg cli

next up

Tanenbaum [1987] Operating Systems

pushf	save flags
cli	disable interrupts
push bp	save the registers
push ax	save ax
push bx	save bx
push cx	save cx
push dx	save dx
push si	save si
push di	save di
push ds	save ds
push es	save es
mov bp,sp	set bp to point to saved es


sp
flags
pushf
sp
flags
cli
flags
sp
bp
push
bp
sp
ax
push
ax
sp
bx
push
bx
sp
cx
push
cx
sp
dx
push
dx
sp
si
push
si
sp
di
push
di
sp
ds
push
ds
sp
es
push
es
sp
bp
mov
bp,sp
sp
bp

clicks offset count

prev next up down

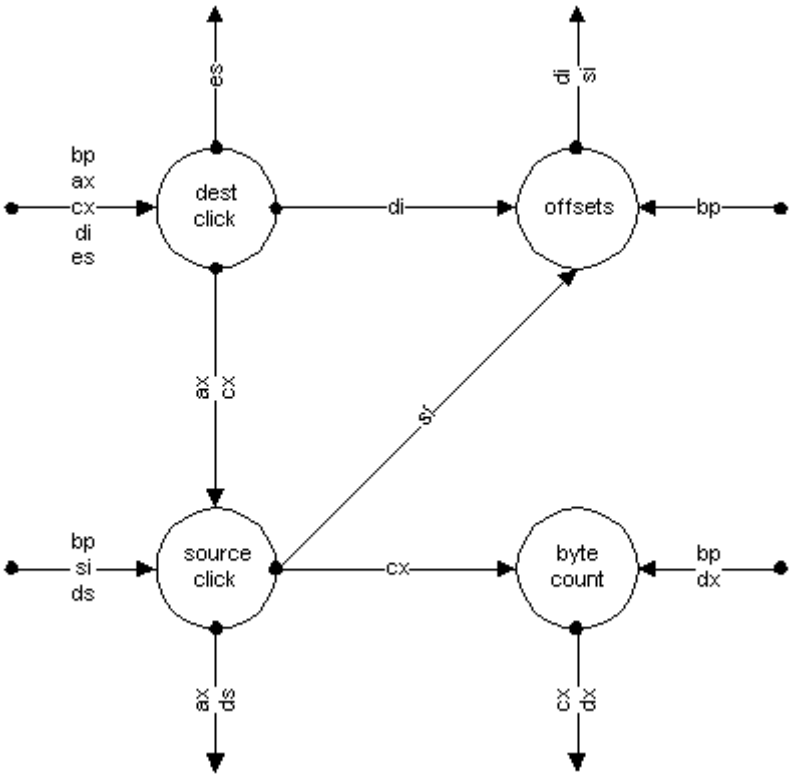
dest click

source click

offsets

byte count

Tanenbaum [1987] Operating Systems

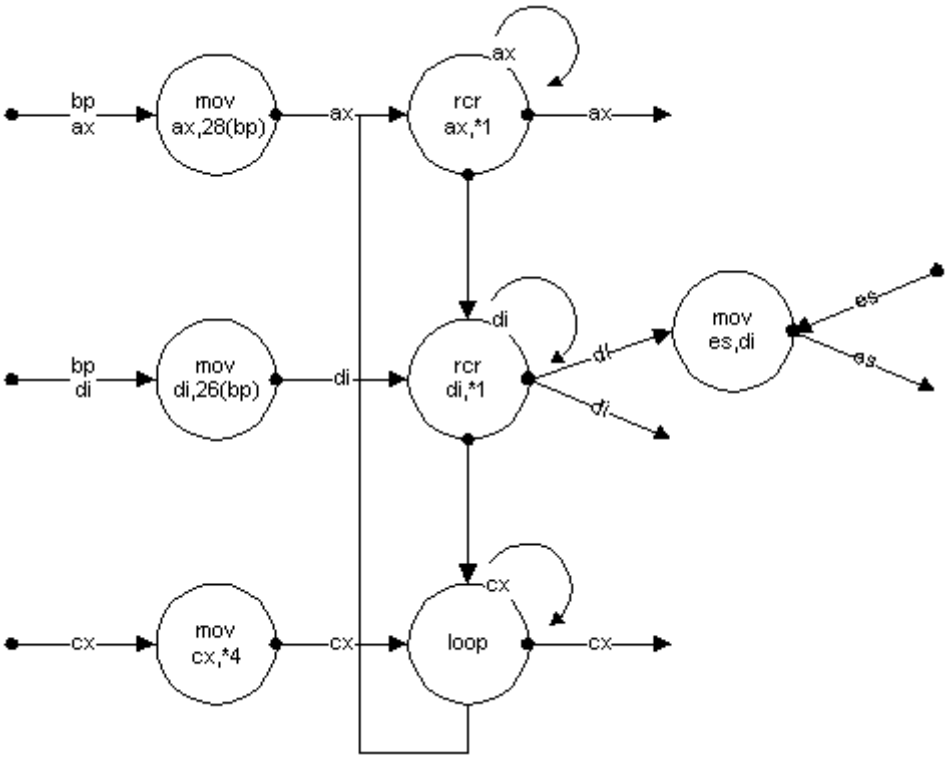


dest click

next up

Tanenbaum [1987] Operating Systems

```
L0:     mov ax,28(bp)     | ax = high-order word of 32-bit destination
        mov di,26(bp)     | di = low-order word of 32-bit destination
        mov cx,*4         | start extracting click number from dest
L1:     rcr ax,*1         | click number is destination address / 16
        rcr di,*1         | it is used in segment register for copy
        loop L1           | 4 bits of high-order word are used
        mov es,di         | es = destination click
```

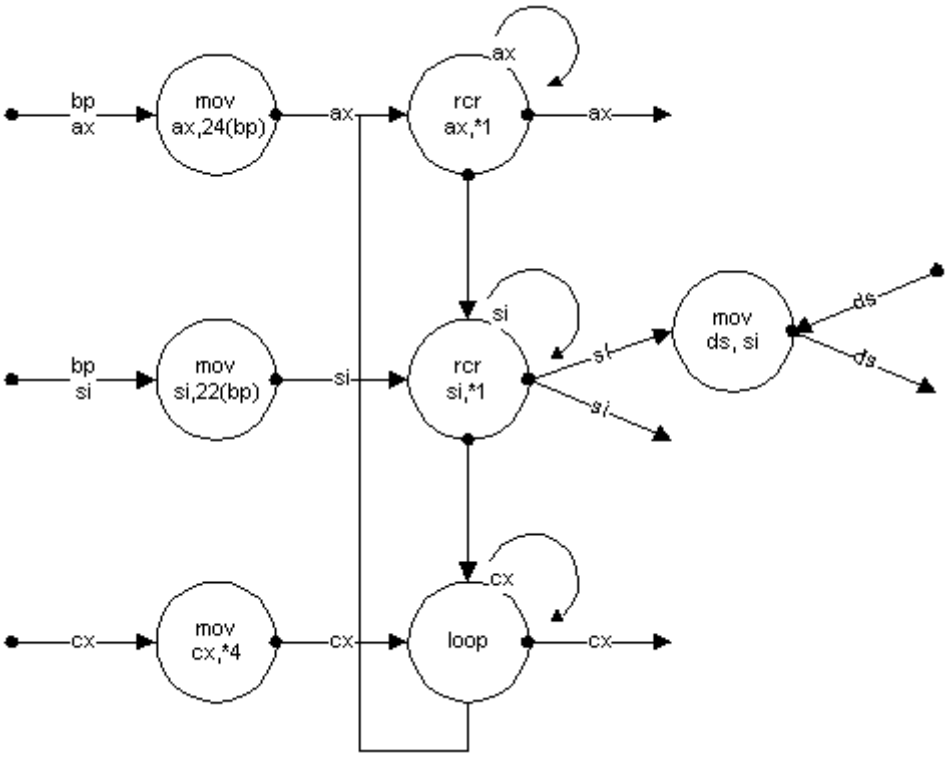


source click

prev next up

Tanenbaum [1987] Operating Systems

```
        mov ax,24(bp)   ax = high-order word of 32-bit source
        mov si,22(bp)   si = low-order word of 32-bit source
        mov cx,*4       start extracting click number from source
L2:     rcr ax,*1       click number is source address / 16
        rcr si,*1       it is used in segment register for copy
        loop L2         4 bits of high-order word are used
        mov ds,si       ds = source click
```

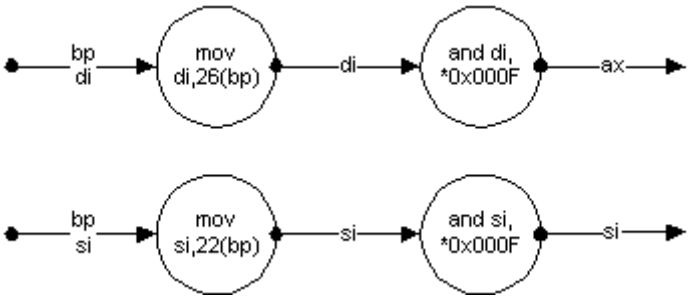


offsets

prev next up

Tanenbaum [1987] Operating Systems

mov di,26(bp)	di = low-order word of dest address
and di,*0x000F	di = offset from paragraph # in es
mov si,22(bp)	si = low-order word of source address
and si,*0x000F	si = offset from paragraph # in ds



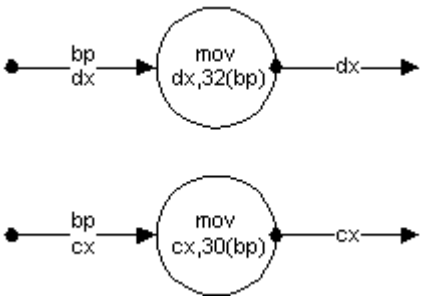
byte count

prev up

Tanenbaum [1987] Operating Systems

```
mov dx,32(bp)
mov cx,30(bp)
```

dx = high-order word of byte count
cx = low-order word of byte count



block copy more next

prev next up down

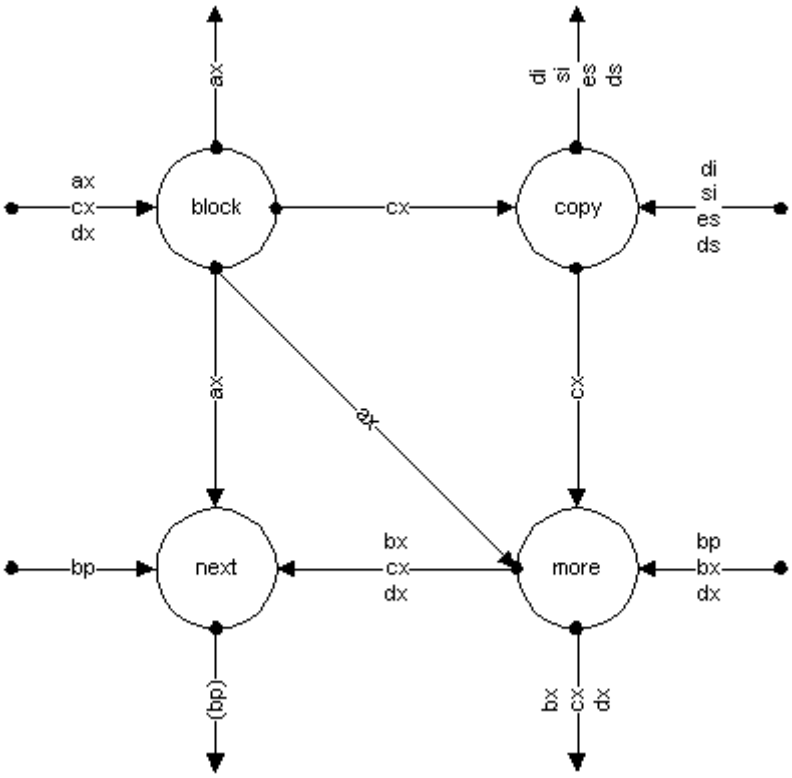
block

copy

more

next

Tanenbaum [1987]
Operating Systems

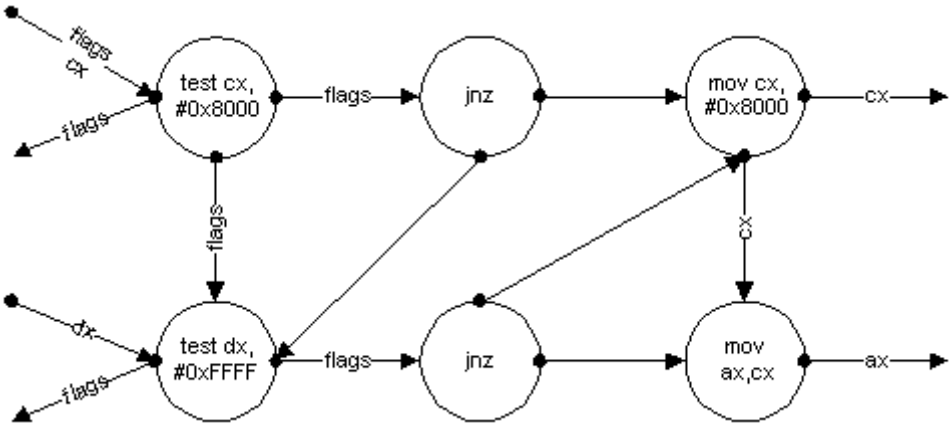


block

next up

Tanenbaum [1987] Operating Systems

```
        test cx,#0x8000     if bytes >= 32768, only do 32768
        jnz L3              per iteration
        test dx,#0xFFFF     check high-order 17 bits to see if bytes
        jnz L3              if bytes >= 32768 then go to L3
        jmp L4              if bytes < 32768 then go to L4
L3:     mov cx,#0x8000      0x8000 is unsigned 32768
L4:     mov ax,ax           save actual count used in ax; needed later
```

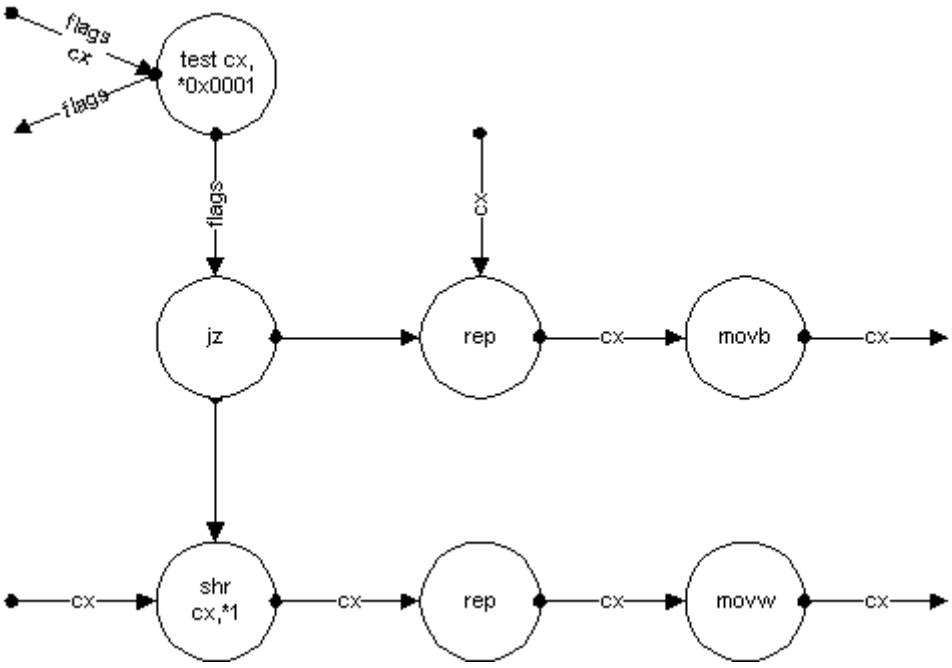


copy

prev next up

Tanenbaum [1987] Operating Systems

```
        test cx,*0x0001     should we copy a byte or a word at a time?
        jz L5               jump if even
        rep                 copy 1 byte at a time
        movb                byte copy
        jmp L6              check for more bytes

L5:     shr cx,*1           word copy
        rep                 copy 1 word at a time
        movw                word copy
```

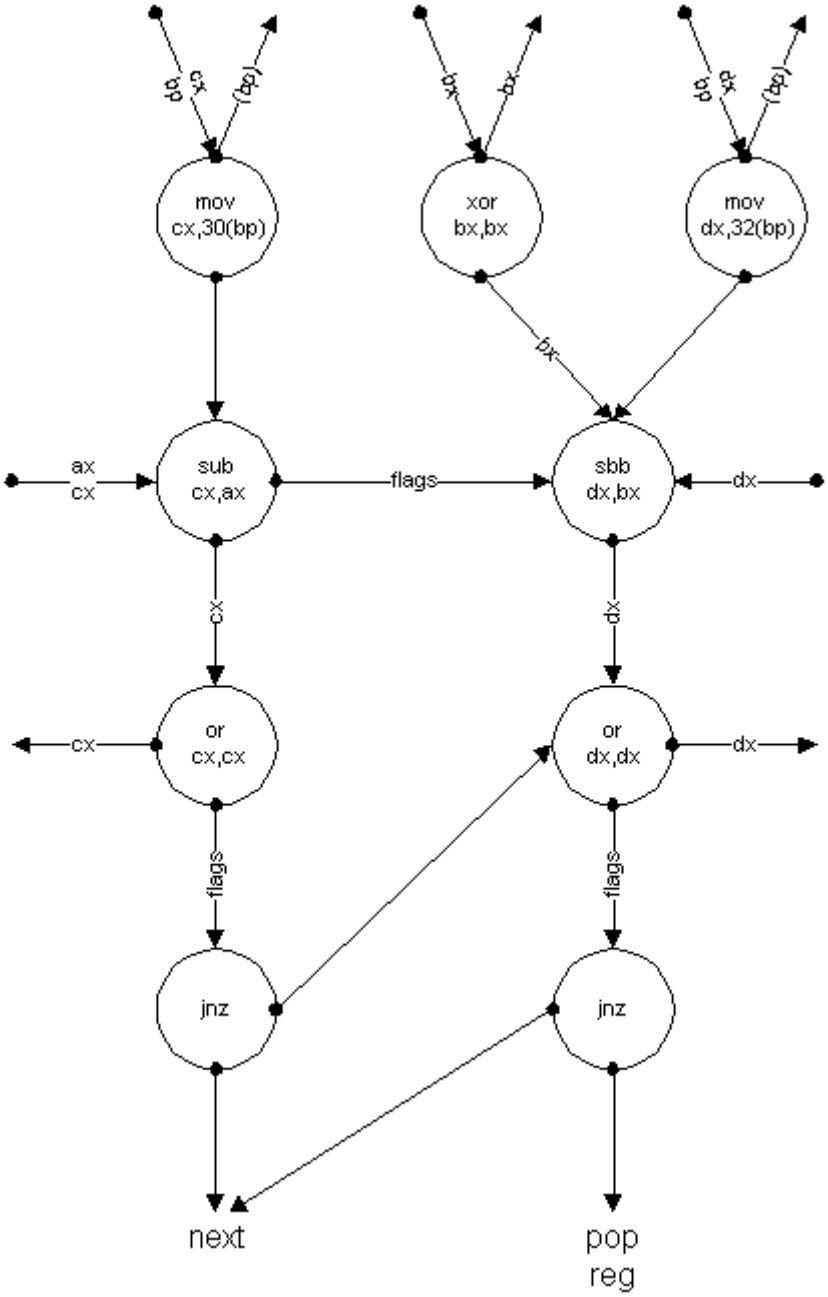


more

prev next up

Tanenbaum [1987] Operating Systems

L6:	mov dx,32(bp)	decr count, incr src & dst, iterate if needed
	mov cx,30(bp)	dx || cx is 32-bit byte count
	xor bx,bx	bx || ax is 32-bit actual count used
	sub cx,ax	compute bytes - actual count
	sbb dx,bx	dx || cx is # bytes not yet processed
	or cx,cx	see if it is 0
	jnz L7	if more bytes then go to L7
	or dx,dx	keep testing
	jnz L7	if loop done, fall through

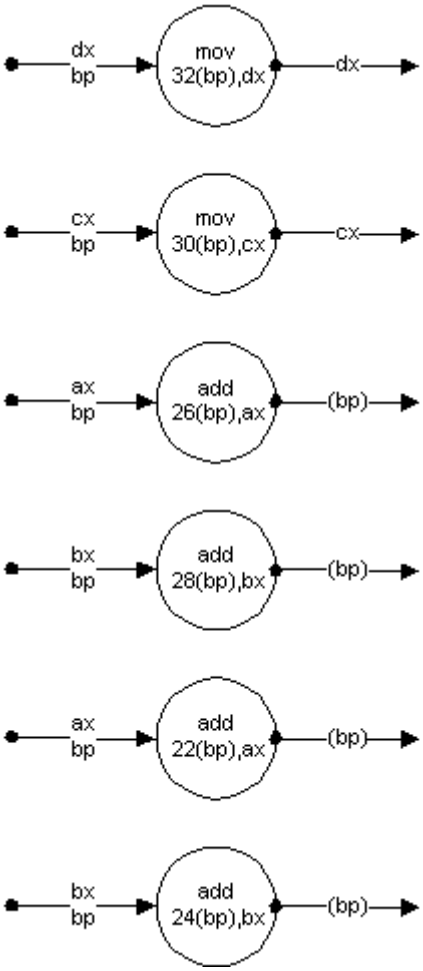


next

prev up

Tanenbaum [1987] Operating Systems

```
L7:     mov 32(bp),dx     | store decremented byte count back in mem
        mov 30(bp),cx     | as a long
        add 26(bp),ax     | increment destination
        adc 28(bp),bx     | carry from low-order word
        add 22(bp),ax     | increment source
        adc 24(bp),bx     | carry from low-order word
        jmp L0            | start next iteration
```

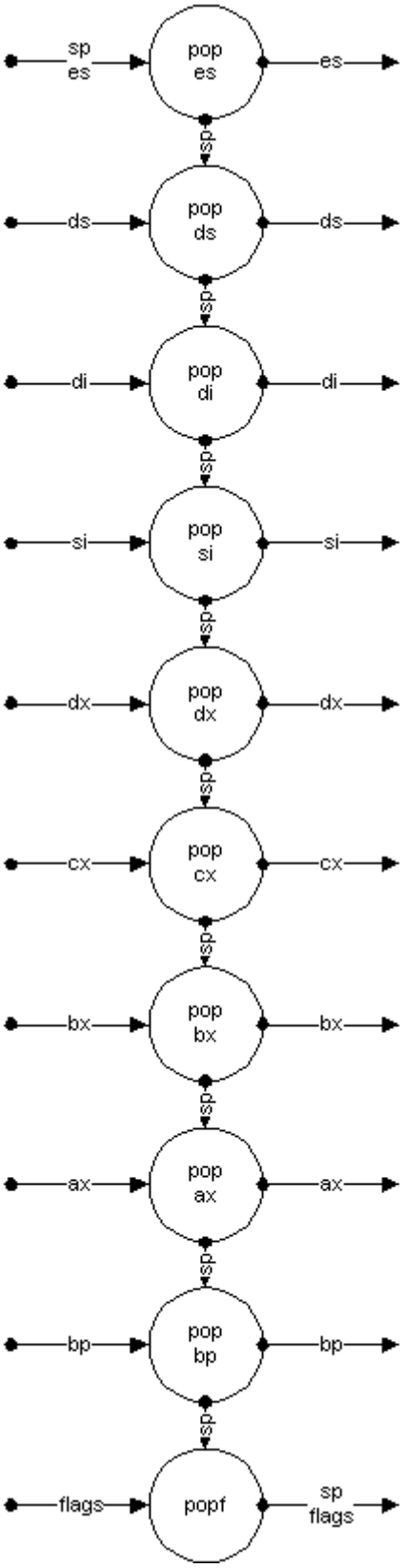


pop reg

prev up

Tanenbaum [1987] Operating Systems

pop es	restore all the saved registers
pop ds	restore ds
pop di	restore di
pop si	restore si
pop dx	restore dx
pop cx	restore cx
pop bx	restore bx
pop ax	restore ax
pop bp	restore bp
popf	restore flags
ret	return to caller


sp
es
pop
es
es
sp
ds
pop
ds
ds
sp
di
pop
di
di
sp
si
pop
si
si
sp
dx
pop
dx
dx
sp
cx
pop
cx
cx
sp
bx
pop
bx
bx
sp
ax
pop
ax
ax
sp
bp
pop
bp
bp
sp
flags
popf
sp
flags