Domain Analysis

Univ.Prof. Dipl.-Ing. Dr. techn.

Harald GALL

Universität Zürich

Technische Universität Wien

http://www.infosys.tuwien.ac.at/Teaching/Courses/SWV/

Inhalt

- Einführung: Definitionen und Begriffe
- Domain Analysis – Konzepte
- Schritte des Domain Engineering Prozesses
  - Rollen im Domain Engineering
- Der Domain Analysis – Prozess
- Feature Oriented Domain Analysis (FODA)
- Zusammenfassung und Ausblick

Domain Analysis - Einführung

- Organisationen entwickeln Software-Systeme, die zahlreiche Gemeinsamkeiten haben
  - im Anwendungsbereich (Telekom-SW, Business SW etc.), oder
  - im technischen Lösungsbereich, oder
  - im Packaging der Software
- Domain Analysis Methoden definieren formale Prozesse, um diese Gemeinsamkeiten herauszulösen und Software Komponenten dafür zu entwickeln
- DA beschäftigt sich mit
  - Identifikation, Beschaffung und Repräsentation von Spezifikations- und Implementierungs-Wissen für eine Klasse von *realen* Problemen
  - der Beschreibung von Komponenten und Architekturen, die für eine Anwendungsdomäne charakteristisch sind

Was ist eine Domäne?

- ...ein Wissensbereich oder Aktivität charakterisiert durch eine Familie verwandter Systeme.
- Eine Domäne wird charakterisiert durch
  - eine Menge von Konzepten und Technologien in einem Wissensbereich [Bass et al. 1997]
  - durch ihren Bereich (Abgrenzung), seine Informationen (Objekte), Features und Verwendungen, sein Verhalten und operationale Charakteristiken [SEI 1999]
  - durch gemeinsame Expertise, gemeinsames Design, und gemeinsamen Markt [Mili et al. 1999]

Was ist Domain Analysis?

- DA ist der Prozess zur Identifikation, Sammlung, Organisation und Repräsentation der relevanten Information in einer Domäne, basierend auf
  - der Analyse existierender Systeme und deren Entwicklungen,
  - Wissen von den Domänen-Experten,
  - zugrunde liegender Theorien und Technologien in einer Domäne [Kang et al. 1999]
- DA ist der Versuch jene Objekte, Operationen und Relationen zu identifizieren, die Domänen-Experten als wichtig für die betreffende Anwendungsdomäne erachten [Neighbors]
- DA = Prozess der Identifikation, Akquisition und Evolution von wiederverwendbarer Information, die im Zuge der Entwicklung von Systemen für Klassen von Applikationen oder Problemdomänen wiederverwendbar werden. [Arango]

Beispiele

- Anwendungsbereich = Flugreservierungssysteme
  - typische Objekte: Sitzplätze, Flüge, Flughäfen, Besatzungen etc.
  - typische Operationen: Reservierung eines Sitzplatzes, Zuweisung einer Crew zu einem Flug etc.
- Anwendungsbereich = Mobiltelefon-Software
  - typische Objekte: incoming/outgoing calls, addresses/contacts ...
  - typische Operationen: calls, sms, mms, accept call ...
- Anwendungsbereich = Music Software
  - …

Grundannahmen

- Wiederverwendbare Information ist anwendungsdomänabhängig
  (domain specifity of reusable information)

- Das Wissen über eine Anwendungsdomäne (*application domain*) ist hinreichend zusammenhängend und stabil, sodass es sich lohnt ein Modell über die Anwendungsdomäne zu bilden
  (cohesiveness and stability of problem domains)

Application Domain

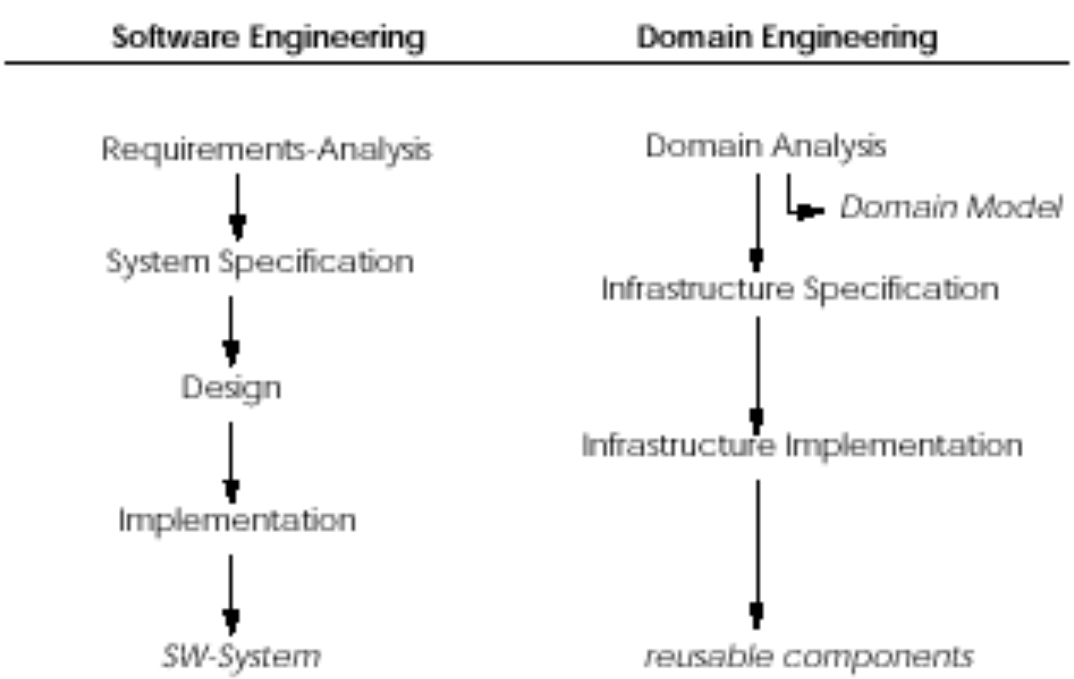
- Eine Menge von Information heißt Application Domain (Anwendungsbereich, Anwendungsdomäne), wenn
  - die Beziehungen zwischen den Objekten hinreichend genau bekannt sind
  - Interesse an der Beschreibung/Lösung von Problemen dieser Art besteht
  - Zugang zu ausreichend Wissen über die zu lösenden Problemen verfügbar ist

- Anwendungsdomäne beschreibt eine Klasse von Problemen
- Vorbedingung für Domain Analysis
  - Definition eines Anwendungsbereichs
  - Abgrenzung des Anwendungsbereichs („scoping“)
  - (erfordert Konsens der daran beteiligten “community”)

Domain Model

- Ziel der Domain Analysis ist die Definition eines Domain Models
- Das Domain Model beschreibt
  - Konzepte, die eine Spezifikation eines Systems aus dieser Domain ermöglichen
  - Vorgangsweisen, wie eine solche Spezifikation eines Systems in Source-Code transformiert werden kann
- Das Domain Model dient als
  - definitive Quelle und Referenz, falls während der Analyse Unklarheiten und Probleme auftreten
  - Repository für das gesammelte Wissen über den Anwendungsbereich
  - Spezifikation für die Implementierung von wiederverwendbaren Komponenten

Domain Engineering



- Wissen über die Problemdomäne ist oftmals implizit und nicht formal
  - → Notwendigkeit dieses Wissen explizit und formal zu repräsentieren
  - → Domain Engineering

Schritte des Domain Engineering

- (1) Domain Analysis
  - Identifizierung, Akquisition und Evolution wiederverwendbarer Information einer Anwendungsdomäne für die Spezifikation und Implementierung von SW-Systemen
    - *conceptual analysis*
      - *Informationen für die Spezifikation von Systemen der Domäne*
    - *constructive analysis*
      - *Informationen für die Implementierung von Systemen der Domäne*

Schritte des Domain Engineering /2

- (2) Infrastructure Specification
  - Selektion und Organisation wiederverwendbarer Information mit Domain Model
  - Spezifikation einer Architektur von wiederverwendbarer Information
    - Klassen von Applikationen, die die Reuse-Infrastruktur implementieren kann, Durchsatz der Reuse-Infrastruktur, ...

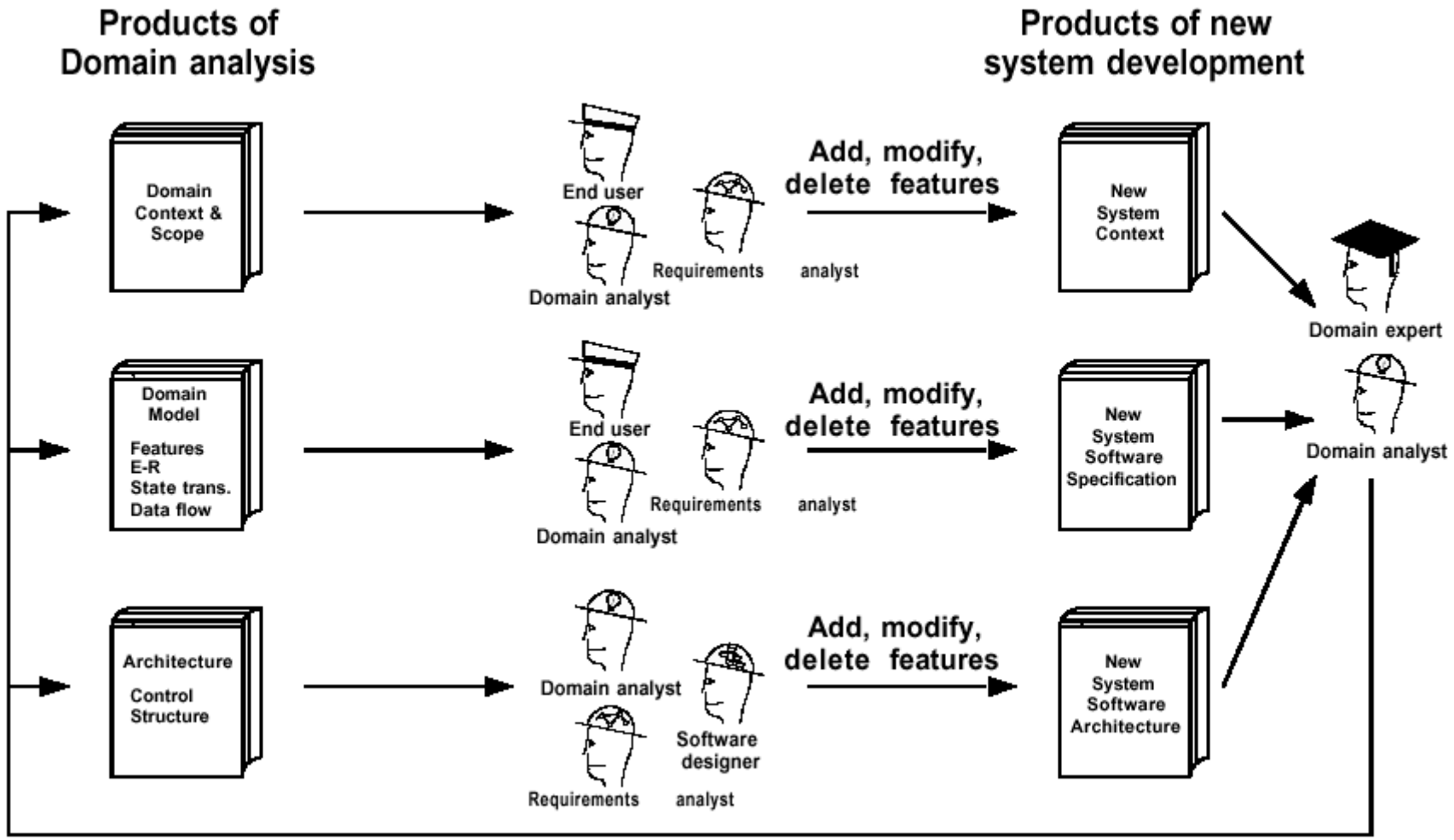
Schritte des Domain Engineering /3

- (3) Infrastructure Implementation
  - Design + Implementierung von wiederverwendbaren Teilen
  - Aktuelle Produktion + Test von Komponenten, Bibliotheks-Strukturen, die die Reuse Infrastructure benötigt, u.dgl.

Domain Analysis Process & Products

- 1) Context Analysis
  - Definition von Umfang und Grenzen einer Domäne
  - erfordert die Repräsentierung von primären inputs & outputs von Software in der Domäne sowie weitere Software Schnittstellen.
- 2) Domain Modelling
  - Beschreibung der Probleme einer Domäne, die durch Software gelöst werden sollen:
    - Features von Software in der Domäne
    - Standard Vokabular von Domain Experts
    - Dokumentation von Entitäten der Software
    - Generische Software Requirements via control/data flow etc.
- 3) Architecture Modelling
  - Erzeugung der Software Architektur(en), die eine Lösung dieser Probleme in der Domäne darstellen
    - Struktur von Implementierungen von Software einer Domäne
    - Repräsentationen in Form von architekturellen Modellen zur Konstruktion und Abbildung vom Domain Model auf die Architekturen

Rollen der DA



Rolle: Domain Expert

- *Aufgaben*
  - Definition und Abgrenzung der relevanten Bereiche eines Anwendungsbereichs
  - Aufsetzen und Abstimmen eines domänen-spezifischen Vokabulars für domänen-spezifische Begriffe
  - Auswahl und Untersuchung einer repräsentativen Auswahl von bestehenden Systemen (Case-Studies)
  - Reviewing und Kritik an den entstehenden Domain Models
- *Personenkreis?*
  - *erfahrene Benutzer von Systemen*
  - *Experten in der Spezifikation und Management in Software-Projekten*
  - *Software-Entwickler bzw. Wartungspersonal aus dem betreffenden Anwendungsbereich*

Rolle: Domain Analyst

- *Aufgaben*
  - Organisation und Führung des “Knowledge-Acquisition”-Prozesses
  - Umsetzung der gewonnenen Information in das Domain Model
  - Verifikation des entstandenen Domain Models in Hinblick auf die verwendeten Repräsentationsmittel
  - Validierung des Domain Models gegenüber existierenden Systemen
  - Untersuchung der Effekte von Änderungen des Domain Models
- *Personenkreis*?
  - erfahrene System-Analytiker mit “knowledge engineering”-Background
  - Fähigkeit zur Abstraktion, Erkennen von Zusammenhängen, Entwicklung von Analogien
  - Kommunikationsfähigkeit mit unterschiedlichsten Personenkreisen (Domain Experten, Anwender, Software-Entwickler, ...)
  - Erfahrungen im betreffenden Anwendungsbereich sind von Vorteil

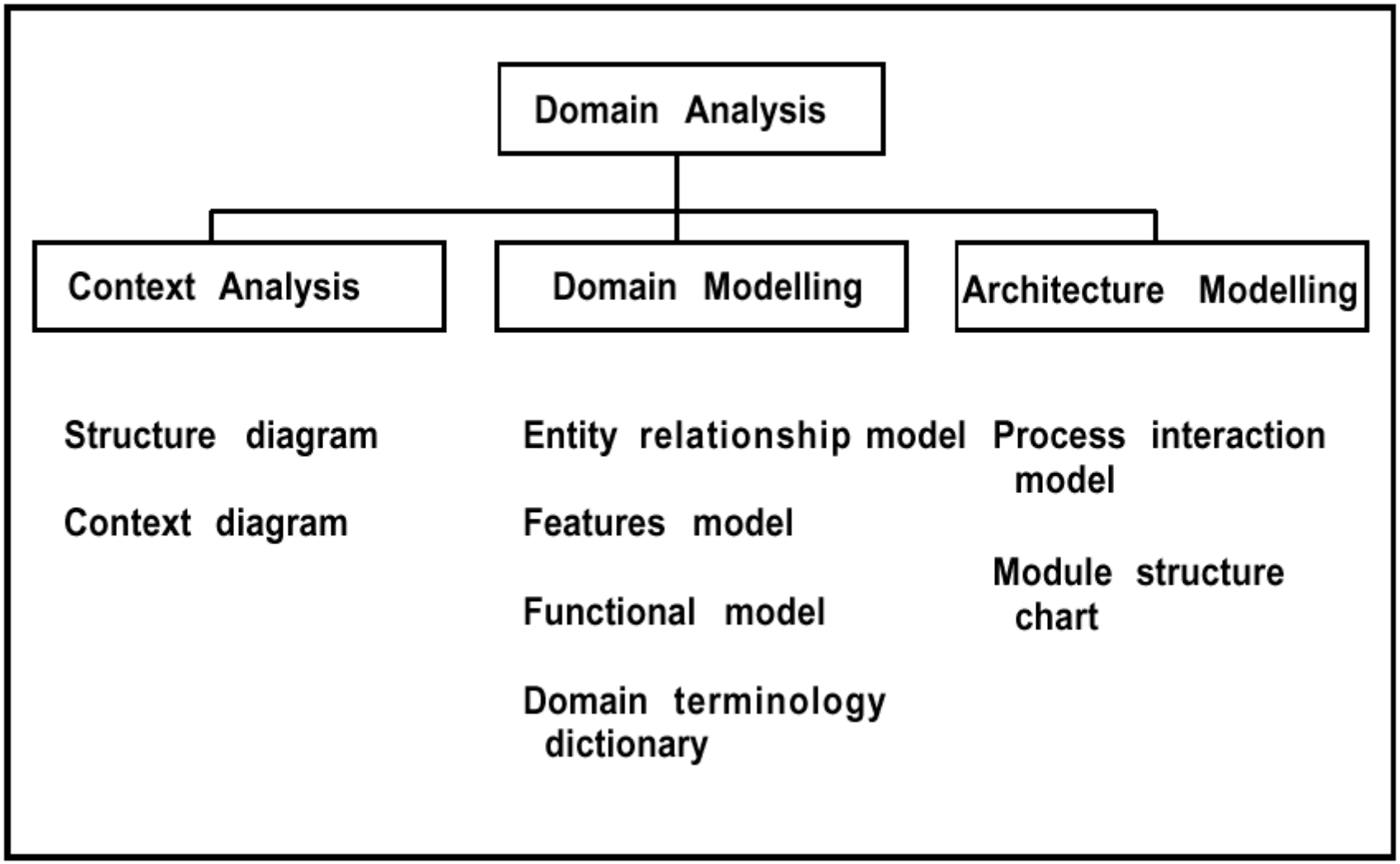
Rolle: Infrastructure Analyst

- *Aufgaben*
  - Definition der Anzahl, Verteilung, Struktur des Repositories für die wiederverwendbaren Informationen
  - Verantwortlich für den Zugriff auf das Repository bzw. die Wartung der dementsprechenden Tools
  - Definition der Komponenten des Repositories bzw. deren Repräsentation
  - Durchführung von Marktanalysen (Abschätzung der Software-Anforderungen der Organisation in einer Anwendungsdomäne, Beobachtung von Kosten/Nutzen Verhältnissen, ...)
- *Personenkreis*?
  - fundiertes Wissen auf dem Gebiet der Software-Entwicklung (Software-Entwickler bzw. System-Analytiker)
  - Wissen über Reuse-Technology
  - Wissen über Software-Engineering Economics (Kostenabschätzungs-Modelle, statistische Methoden, u.dgl.)

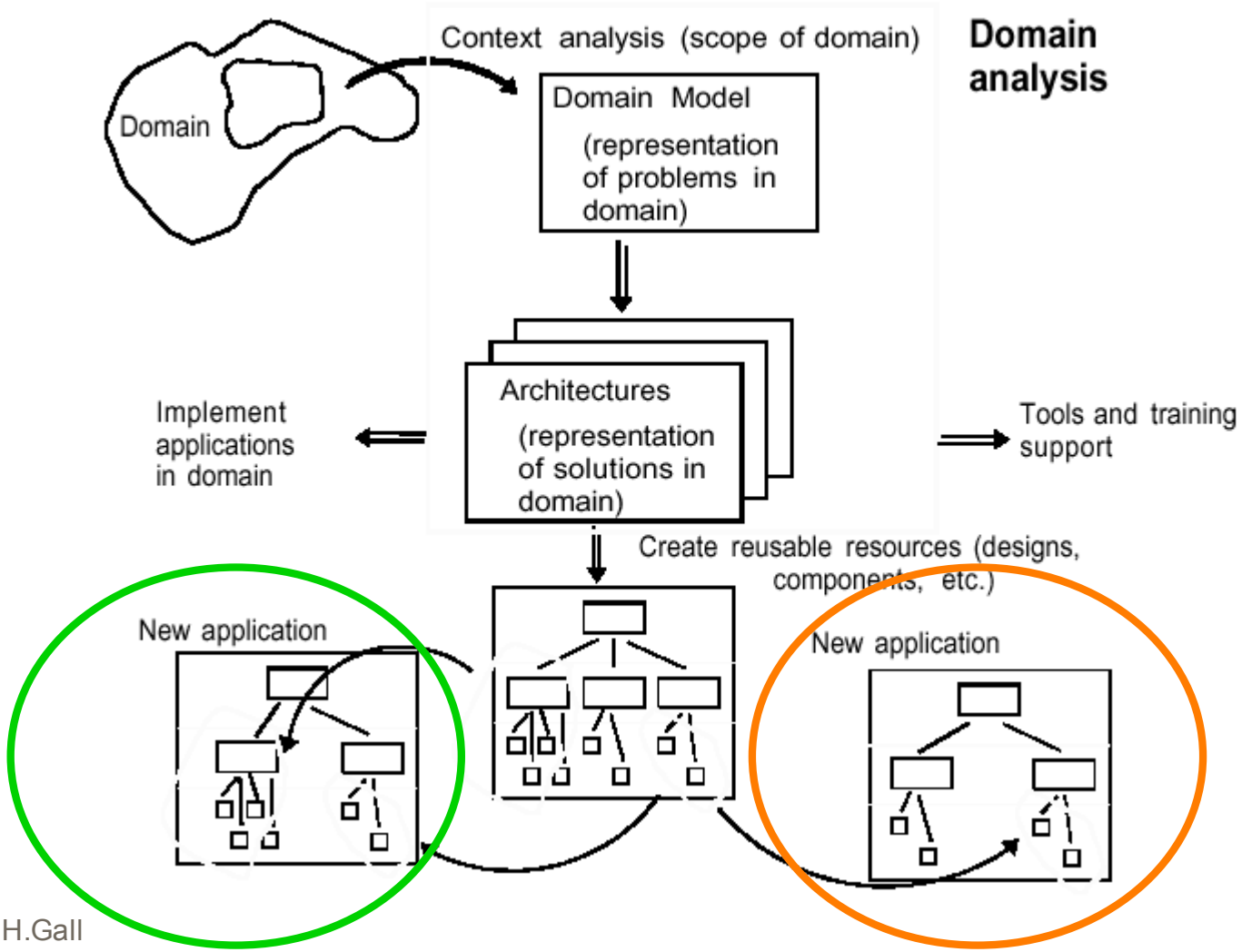
Rolle: Infrastructure Implementer

- *Aufgaben*
  - Entwicklung (+ Testen, Dokumentieren, Warten, ...) der wiederverwendbaren Komponenten und Zugriffs-Mechanismen
  - Organisation der Datenbanken und Index-Strukturen
  - Software-Entwickler der Reuse-Infrastruktur
- *Personenkreis*?
  - Software-Designer bzw. -Entwickler mit herkömmlicher Ausbildung (Fähigkeiten) aber fundierter Einschulung in die Reuse Technology bzw. die Verwendung der Reuse-Infrastruktur

Phasen und Produkte der DA



DA supports Software Development



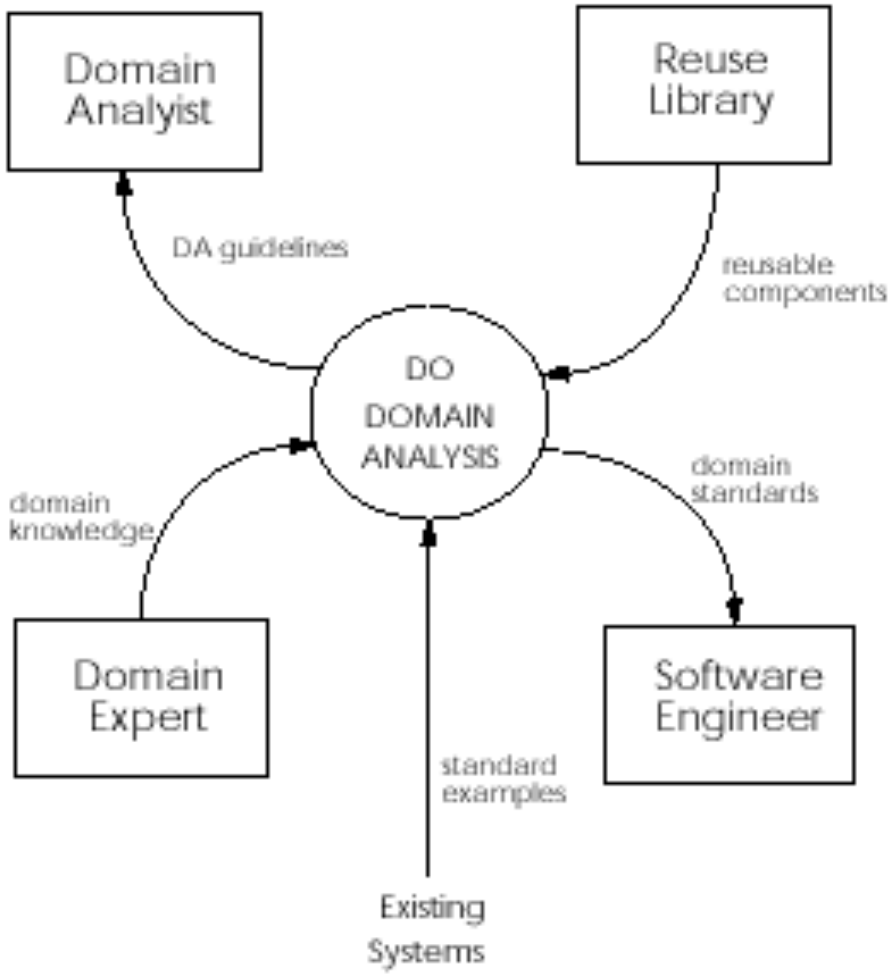
Domain Analysis - Ansätze

- Feature Oriented Domain Analysis (FODA), SEI-CMU
- DA nach Prieto-Diaz
- Organization Domain Modeling (ODM) [Simos 1995]
- Joint Object-Oriented Domain Analysis (JODA) [Holibugh 1982]
- Reuse Library Process Model (RLPM) [RLPM 1991]
- Domain Analysis and Design Process (DADP) [SofTech 1993]
- Domain-Specific Software Architecture (DSSA) [Braun 1992]
- SYNTHESIS [SPC 1993]
- Reuse Business Methodology [Jacobson et al. 1997]

Domain Analysis nach Prieto-Diaz

- DA-Ansatz von Prieto-Diaz besteht aus mehreren Phasen:
  - 1) Pre-DA activities
    - Definition und Abgrenzung des application domain
    - Identifikation der Wissensquellen über die Domäne
  - 2) DA activities
    - Identifikation der Objekte und Operationen
    - Abstraktion
    - Klassifikation für die Abspeicherung im Repository
  - 3) Post-DA activities
    - Encapsulation (= Identifikation und Implementierung der wiederverwendbaren Komponenten, z.B. Interface-Definitionen, Constraints, u.dgl.)

Domain Analysis nach Prieto-Diaz /2



Feature-Oriented Domain Analysis (FODA)

- Supports reuse at the functional and architectural levels
- Domain products represent the common functionalities and architecture of applications in a domain
- Specific applications may be developed as *refinements* of the domain products
- DA is related to requirements analysis and high-level design, but performed in a much broader sense and generates different results
- Encompasses a *family of systems* in a domain, rather than a single system
- Producing a domain model with parameterization to accomodate differences and a standard architecture for developing software components
- Ideal domain model would be applicable throughout the life-cycle from requirements analysis through maintenance

FODA Modelling Primitives

- Aggregation/decomposition
- Generalization/specialization [Borgida85]
- Parameterization [Goguen 84]
  - FODA applies aggregation and generalization primitives to capture commonalities of applications in the domain
  - Differences between applications are captured in refinements
  - Parameters uniquely specify the context of each specific refinement

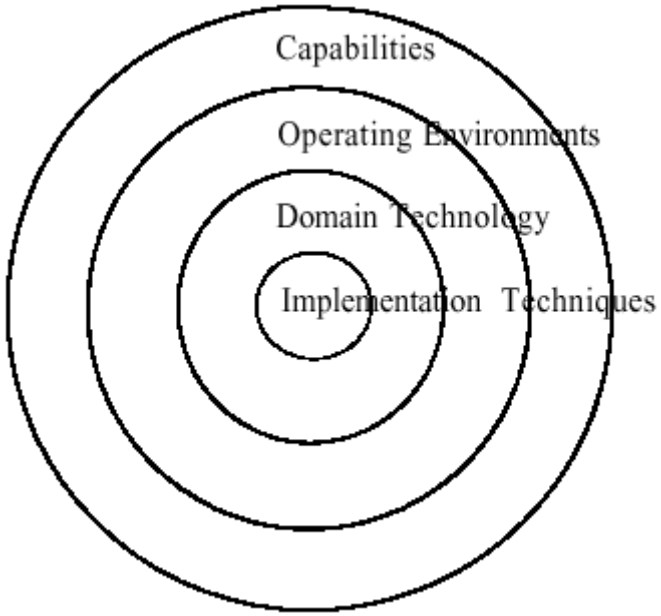
- → Domain Product consisting of a collection of abstractions and a series of refinements of each abstraction with parameterization.

- → Every *new refinement* must be defined in context and parameters before being incorporated in the domain product.

- → FODA domain product is not the end-product (*evolves, expands*)

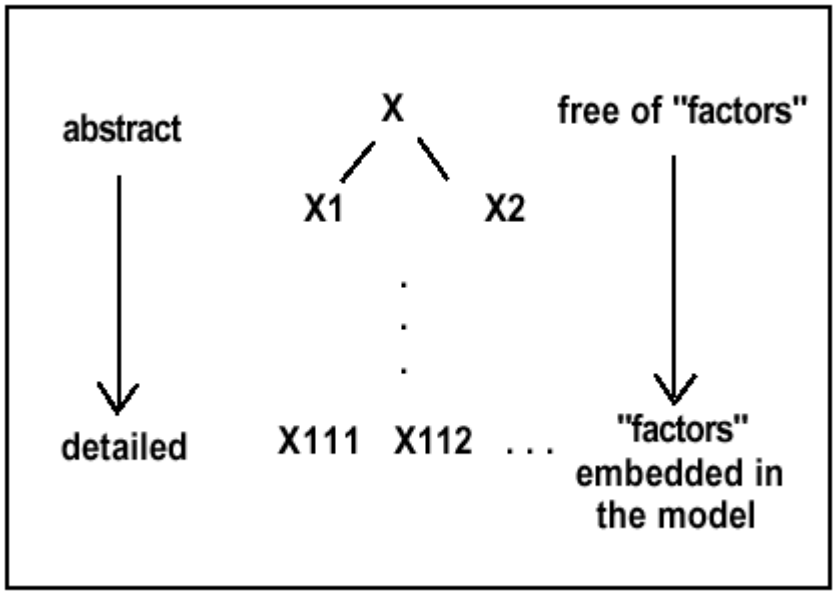
FODA Product Parameterization

- FODA applies this principle also to high-level design
- Factors that result in different applications in a domain:
  - capabilities of applications (end-user‘s perspective)
  - operating environments for applications
  - application domain technology based on which requirements decisions are made
  - implementation techniques (e.g. synchronization used in design etc.)



FODA Model Abstraction

- „x“ represents commonalities of all applications considered in the domain analysis
- as „x“ is refined, info on the context in which each refinement is made is incorporated into the refinement
- „x1“ and „x2“ are more context sensitive and less reusable than „x“
- both providing generic components and refinements of those at various levels
- As generic components are refined, the factors that make applications unique are incorporated in the refinements.



FODA Model Reuse

level of "factors" incorporated in a model	level of abstraction	level of reuse potential	level of productivity increase
generic (free of "factors," i.e., context free)	high	high	low
↓			
application specific ("factors" fully incorporated, i.e., context sensitive)	low	low	high

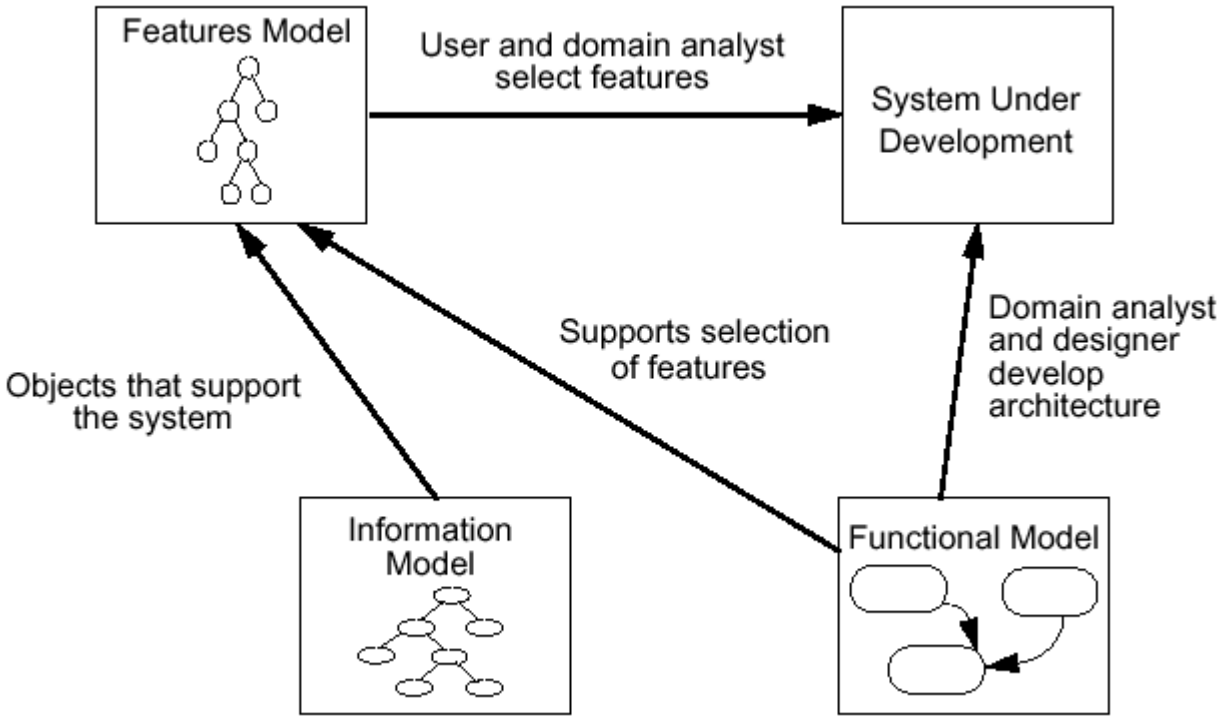
DA Information Sources

Source	Advantages	Disadvantages
Textbooks	• Good source of domain knowledge, theories, methods, techniques, models	• Reflects only specific author's views • May use idealized or biased models
Standards	• Represents standard reference model for domain	• Model may not be current with new technology
Existing Applications	• Most important source of domain knowledge • Can be directly used to determine user-visible features • Requirements documents available for domain model • Detailed design & source code show architectures	• Cost of analyzing many systems is high
Domain Experts	• Can provide contextual/rationale information unavailable elsewhere • Can be consultant during DA, validator of products afterwards	• Experts have different areas of expertise; several experts may be needed

FODA Activities and Products

Phase	Inputs	Activities	Products
Context Analysis	Operating environments, Standards	Context analysis	Context model
Domain Modeling	Features, Context model	Features analysis	Features model
	Application domain knowledge	Information modeling	Information model
	Domain technology, Context model, Features model, Information model, Requirements	Functional analysis	Functional model
			Behavioral model
Architectural Modeling	Implementation technology, Context model, Features model, Information model, Design information	Architectural modeling	Structured executive
			Subsystems model(s)

Use of Domain Model in System Development



FODA Context Analysis

- *Purpose of context analysis* is to define the scope of a domain that is likely to yield exploitable domain products.
  - relationships between candidate domain and elements external to the domain are analyzed
  - variability of relationships and external conditions (e.g., different applications using the domain and their data requirements, different operating environments, etc.) are evaluated.
- *Results of the context analysis*, along with other factors such as availability of domain expertise,domain data, and project constraints, are used to scope the domain.
- Final results of the context analysis are documented in a context model.
  - defines the boundary of the domain, that is, the scope of the domain modelling which follows the context analysis.

Context Model Description

- For each entity in the context model, the following information should be included in the document.
  - Name of the entity (an object on the diagram).
  - Description of the function for a functional entity or description of the contents for a data entity.
  - Applicable standards and/or reusable components.
  - Description of variability, including the range, frequency, and binding time (i.e., compile-time, activation-time, and runtime) of the variation for details of feature binding times).
  - Other items describing the attributes of the entity.
  - Source for the information or for additional information.

Context Model Process

- 1. Make an initial "cut" of the target domain and the domain boundary. (It is assumed that a candidate domain for context analysis is already selected.) Identify the existing applications in the domain or applications using the domain. Identify information sources and any previous works on the domain including domain analysis products, standards, and books and technical papers.
- 2. For each application identified for the analysis, describe the context of the candidate domain in terms of structure diagrams and data-flow diagrams. Verify that the domain has a clear physical boundary within the application. If there are any previous domain analysis results or standards, determine if they address each application adequately; record problems, if any.
- 3. Understand the usage of the target domain, any recent technical developments that will affect the domain, and any future plans or requirements.
- 4. Analyze the variability of the usages and the contexts of each use.
  - a. Based on the features of applications in the domain, begin the definition of a feature model.
  - b. Identify the environmental differences (i.e., operating environments).
  - c. Identify any assumptions (sub-domains and standards) on which the target domain is based.
  - d. Construct a common model by classifying specifics of the contexts into general categories so that each context can be defined as an instantiation of the common model.

Context Model Process /2

Context Analysis

- 5. Evaluate the model against the applications used in the analysis and incorporate the variability information (i.e., how the common model can be adapted to each context) into the context model.
- 6. Evaluate the commonality of the applications, feasibility of constructing domain products, and potential uses and benefits of the domain products.
- 7. Estimate the resources for the subsequent activities considering availability of domain experts, previous work, and maturity of the domain.
- 8. Document the context model; define the terms used in the model in the domain terminology dictionary.
- 9. Validate the model against at least one application that is not included in the analysis. Also, have the model reviewed by domain experts. Record the validation results in the context model documentation.

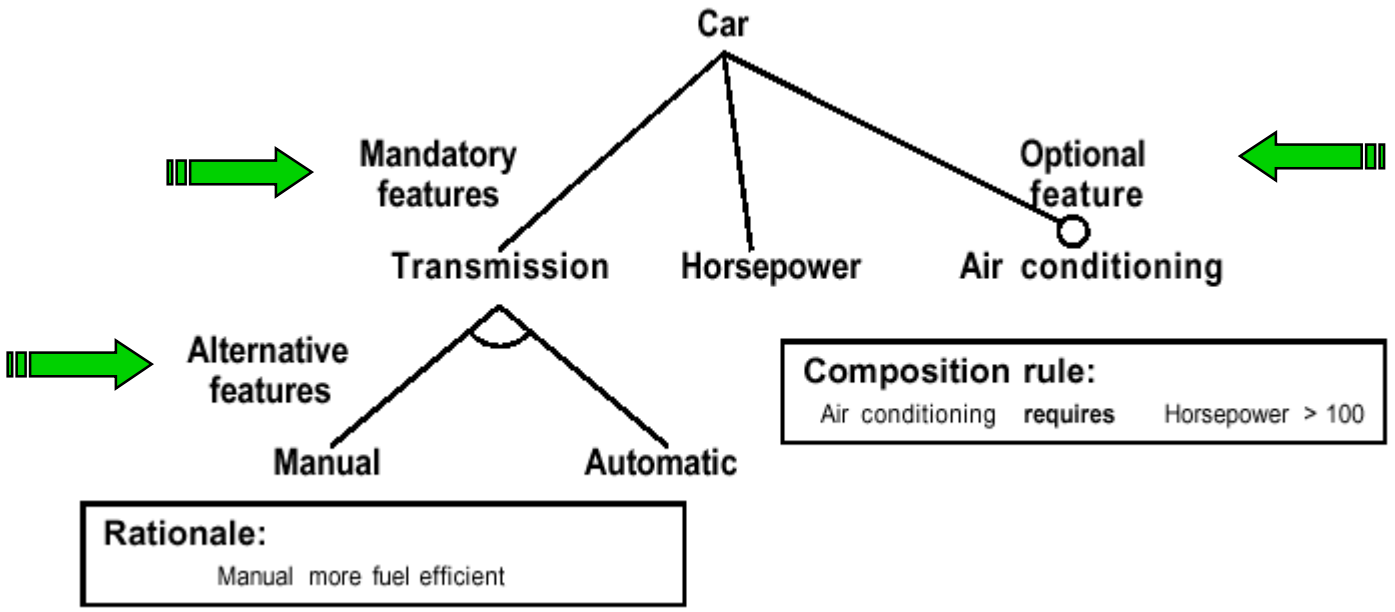
FODA Activities and Products

Phase	Inputs	Activities	Products
Context Analysis	Operating environments, Standards	Context analysis	Context model
Domain Modeling	Features, Context model	Features analysis	Features model
	Application domain knowledge	Information modeling	Information model
	Domain technology, Context model, Features model, Information model, Requirements	Functional analysis	Functional model
			Behavioral model
Architectural Modeling	Implementation technology, Context model, Features model, Information model, Design information	Architectural modeling	Structured executive
			Subsystems model(s)

Domain Modeling / Feature Analysis

- The purpose of feature analysis is to capture in a model the end-user's (and customer's) understanding of the *general capabilities* of applications in a domain.These capabilities might include:
  - services provided by the application,
  - performance of the application,
  - hardware platform required by the application,
  - cost,
  - others
- Feature model should capture the common features and differences of the applications in the domain. The features in the feature model will be used to generalize and parameterize other models.

Features of a car



Feature Model

- A feature model represents the *standard* features of a family of systems in the domain and relationships between them
- structural relationship *consists of* (a logical grouping of features)
- *Alternative* or *optional* features of each grouping must be indicated in the feature model
- Composition rules define the semantics existing between features that are not expressed in the feature diagram. All optional and alternative features that *cannot* be selected when the named feature is selected must be stated using the *"mutually exclusive with"* statement.
- All optional and alternative features that *must* be selected when the named feature is selected must be defined using the *"requires"* statement.
- Selection of optional or alternative features is made based on a number of objectives or concerns that the end-user (and customer) has.

When to „bind“ or „fix“ a feature

- Compile-time features
  - features that result in different packaging of the software and, therefore, should be processed at compile-time.
  - Examples: those that result in different applications (of the same family), or those that are not expected to change once decided (time and space efficiency).
- Load-time features
  - features that are selected or defined at the beginning of execution but remain stable during the execution
  - e.g. features related to the operating environment such as terminal types
- Runtime features
  - features that can be changed interactively or automatically during execution
  - e.g. menu-driven software

Feature Analysis Process

- (1) Source documents collection
- (2) Feature identification
- (3) Feature abstraction, classification, and modelling
- (4) Feature definition
- (5) Model Validation

Feature Identification

- General application features
  - *operating environments, capabilities, domain technology, implementation techniques*
- FODA focuses on capabilities
  - *functional features*
    - basically "services" that are provided by the applications → can be found in user manual and the requirements documents
  - *operational features*
    - those that are related to the operation of applications (again, from the user's perspective); how user interactions with the applications occur → user manuals
  - *presentation features*
    - those that are related to what and how information is presented to the users → user manuals and requirements documents

Feature Abstraction, Classification, and Modelling

- A hierarchical model should be created by classifying and structuring features using the *consists-of* relationship.
- Whether or not each feature is *mandatory*, *alternative*, or *optional* should be indicated in the model.
- Each feature in the model should be defined. The description should also indicate whether it is a *compile-time*, an *activation-time*, or a *runtime feature*.
- The classification of the features can be *used in the components construction* for modularization and for selection of appropriate development techniques.

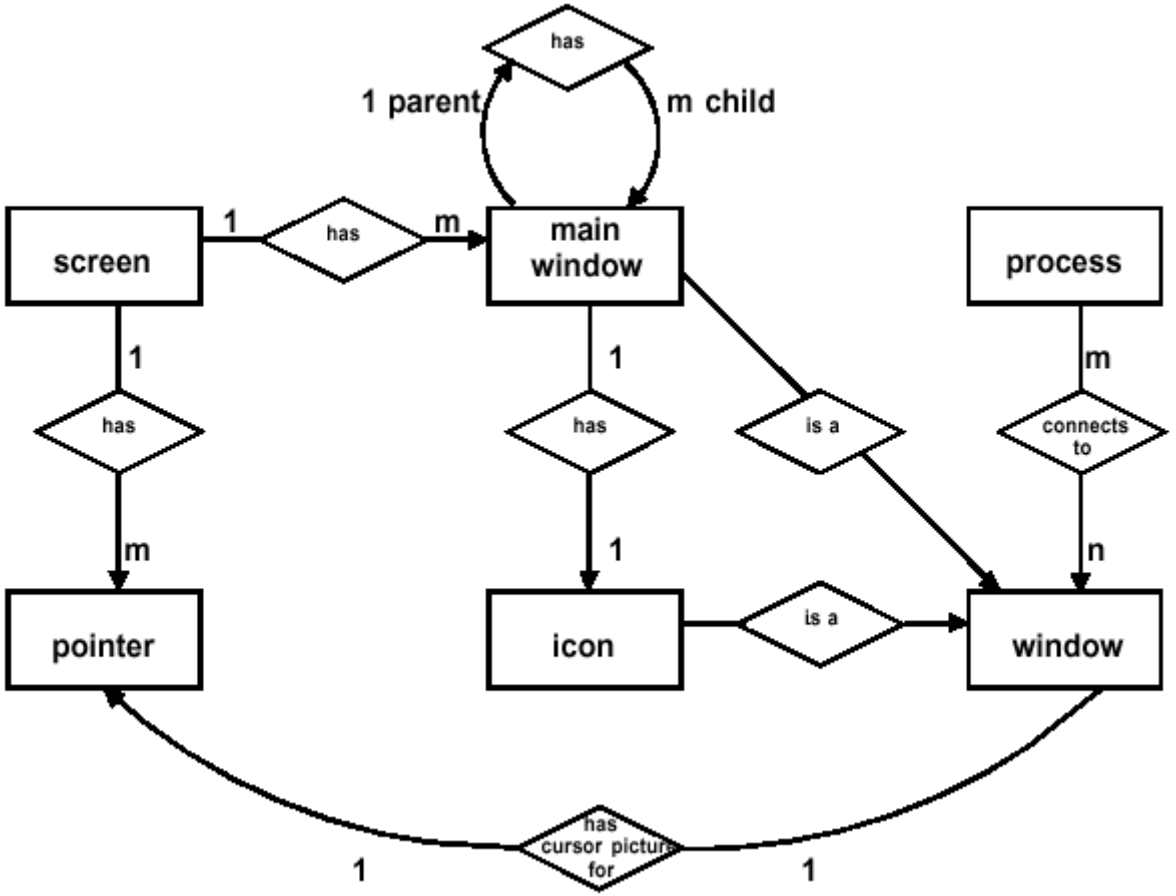
Model Validation

- Whether the feature model correctly represents the features of the domain must be validated by domain experts and against existing applications.
- Domain experts of analysis should not validate model
- A new *set* of several applications would provide a better validation (not always possible due to maturity of domain)

Information Modeling

- Purpose
  - to represent the domain knowledge explicitly in terms of domain entities and their relationships, and to make them available for the derivation of objects and data definitions during the functional analysis and architecture modelling.
    - contextual information which gets lost after the development
    - information that is deeply embedded in software and is often difficult to trace
- Model
  - Entity-relationship modelling technique [Chen 76])
    with semantic data modelling [McLeod 78], [Borgida 84]
  - Chen's notation and method are used, with the adoption of the generalization and aggregation concepts from semantic data modelling that are used as predefined relationship types.
  - Basic building blocks are entity classes and *consist-of* and *is-a* relationships.
    - Entity classes represent abstractions of objects in the application domain and the relationship types *is-a* and *consists-of* represent generalization and aggregation relationships
    - Aggregation relationships specify composition structures between entities
    - Generalization relationships specify commonalities and differences among entities

Example: Window Manager ERD

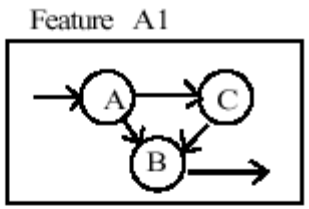


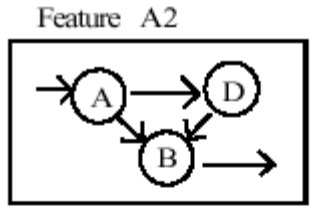
Functional Analysis

- Functional analysis identifies *functional* commonalities and differences of the applications in a domain.
- It abstracts and structures the common functions in a model from which application-specific functional models can be instantiated or derived with appropriate adaptation.
- The mandatory features and the entities are the basis for defining an abstract functional model. The alternative and optional features are embedded into the model during refinement.
- Also, any factors (other than features) that cause functional differences between the applications are defined as *issues* and *decisions*, which are used in the refinements for parameterization.

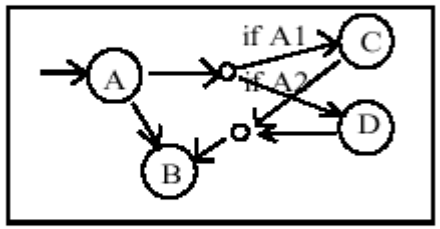
Parameterization through features and issues/decisions

- By developing separate components (refinements) for every alternative: Case 1
- By developing one component, but with parameterization for adaptation to each alternative: Case 2
- By defining a general component and developing each alternative as an instantiation of the general component (with an inheritance mechanism) [Borgida 84]: Case 3





Case 1



or

Activity AA:

perform A,B
if A1 perform C
if A2 perform D

Case 2

Activity AA:
perform A, B

Activity A1:
is-a AA
perform C

Activity A2:
is-a AA
perform D

Case 3

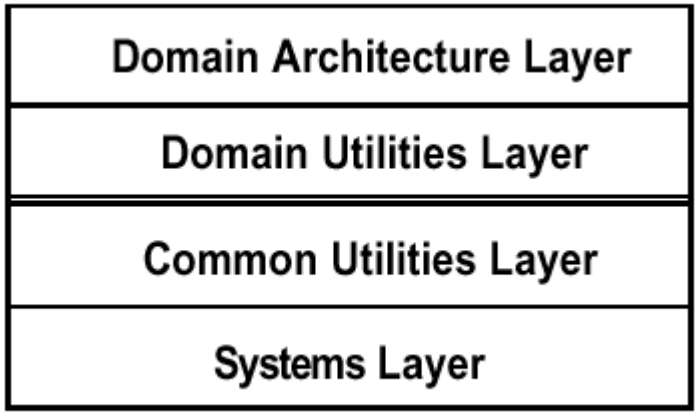
FODA Activities and Products

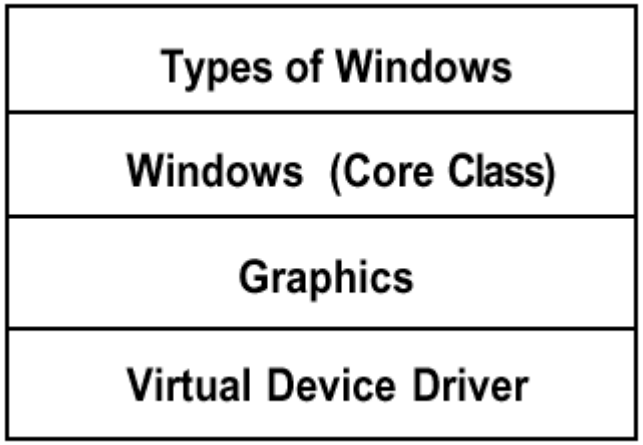
Phase	Inputs	Activities	Products
Context Analysis	Operating environments, Standards	Context analysis	Context model
Domain Modeling	Features, Context model	Features analysis	Features model
	Application domain knowledge	Information modeling	Information model
	Domain technology, Context model, Features model, Information model, Requirements	Functional analysis	Functional model
			Behavioral model
Architectural Modeling	Implementation technology, Context model, Features model, Information model, Design information	Architectural modeling	Structured executive
			Subsystems model(s)

FODA Architectural Modeling

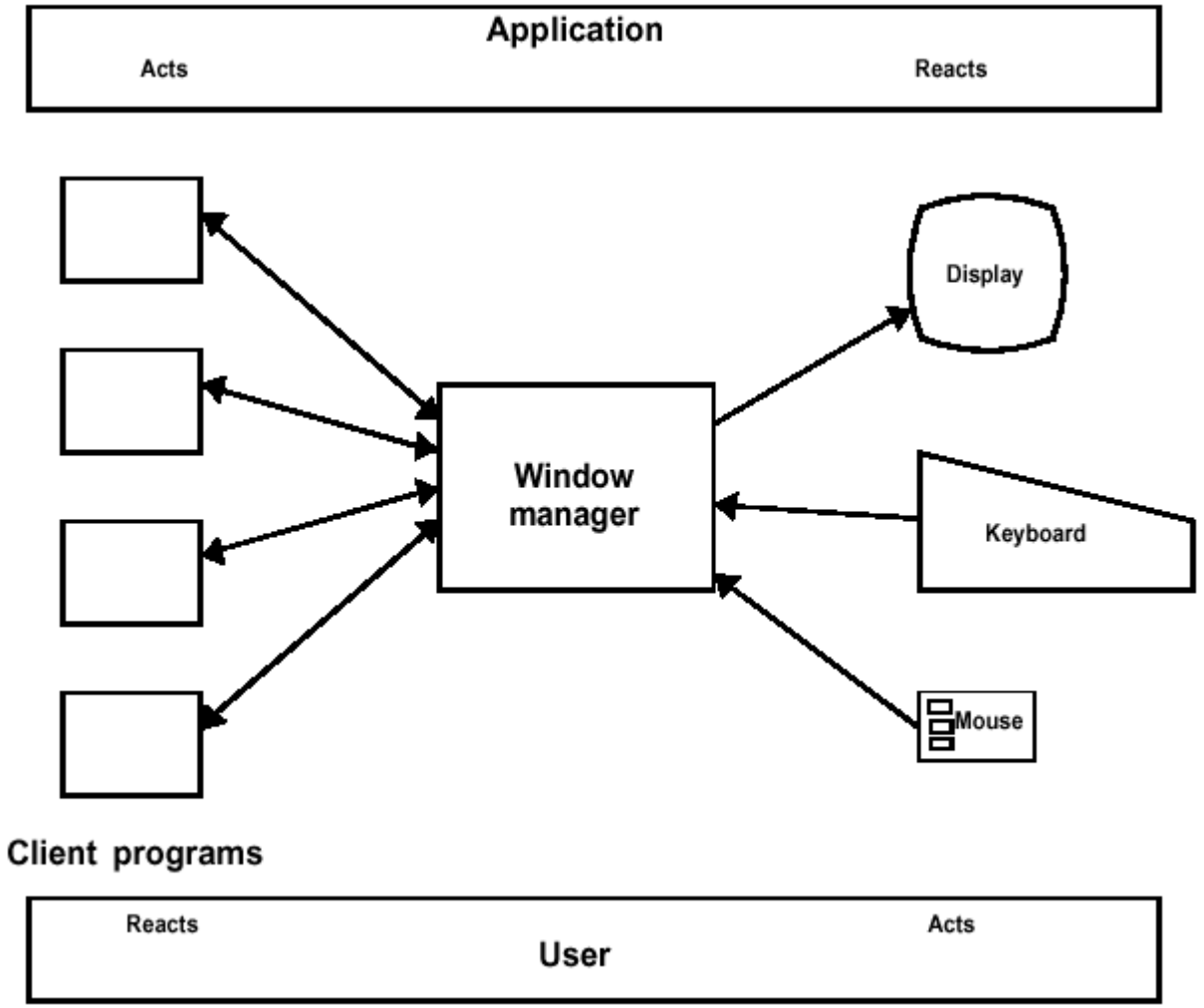
- Purpose:
  - to provide a software "solution" to the problems defined in the domain modelling phase. An architecture model (also known as a *design reference mode*l) is developed, and from it detailed design and component construction can be done.
- A FODA architecture model is a high-level design of the applications in a domain
  - focuses on identifying concurrent processes and domain-oriented common modules, and
  - on allocating the features, functions, and data objects defined in the domain model to the processes and modules.

Architectural layers





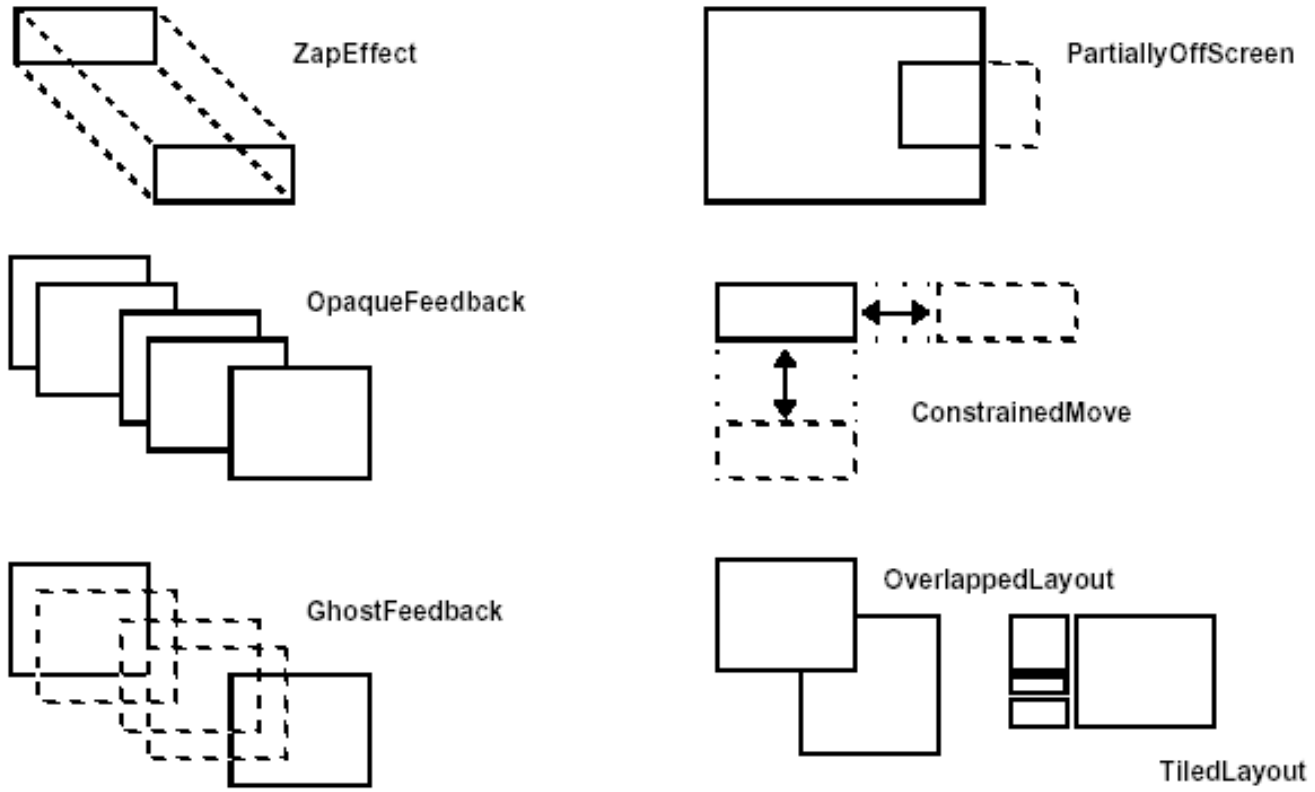
Example: Window Management System (WMS)



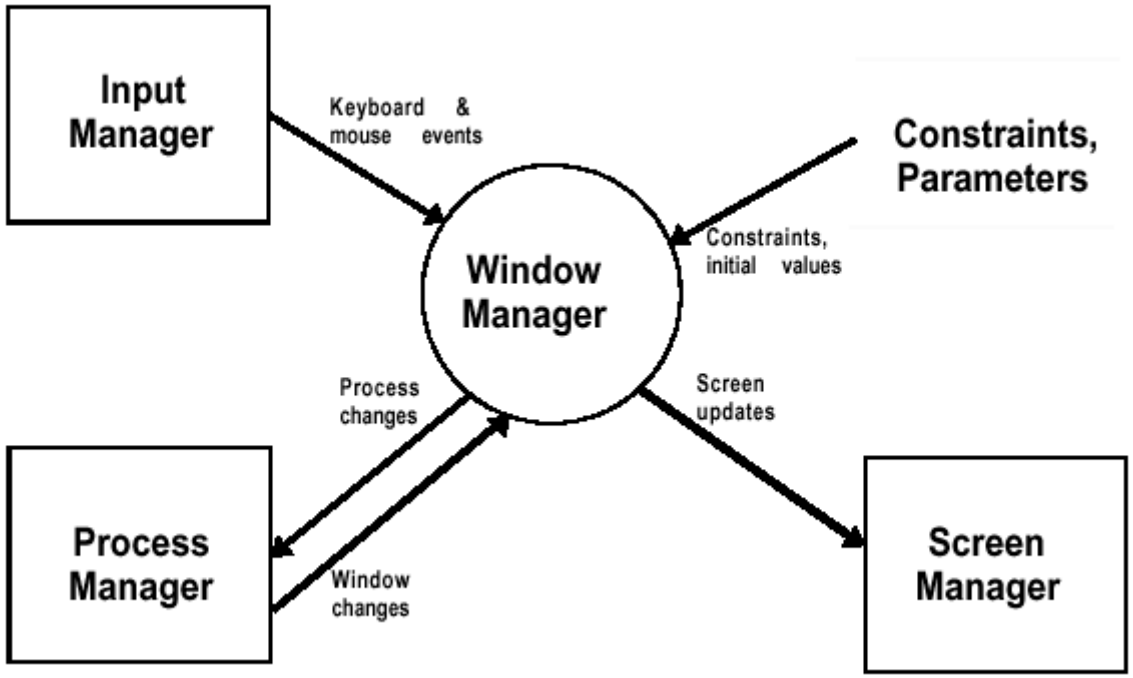
WMS capabilities

- Capabilities
  - Create and Destroy
  - Move and Resize
  - Iconify and Deiconify
  - Hide, Expose, Circulate
  - Change Focus
  - Refresh
- e.g. Features of „Move“
  - `constrainedMove`
  - `eraseBefore/eraseAfter`
  - `exposeAfterMove`
  - `ghostFeedback/opaqueFeedback`
  - `moveIcon`
  - `...`

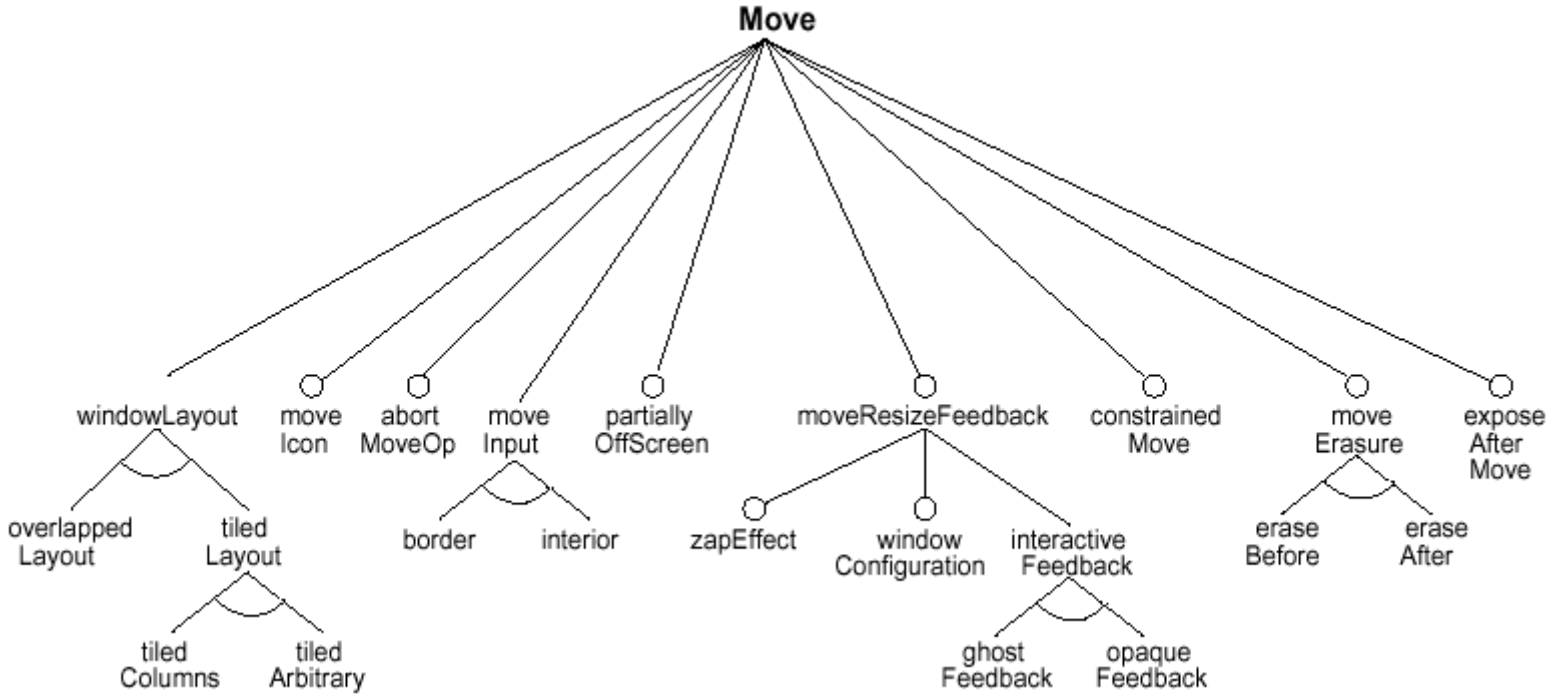
WMS user features



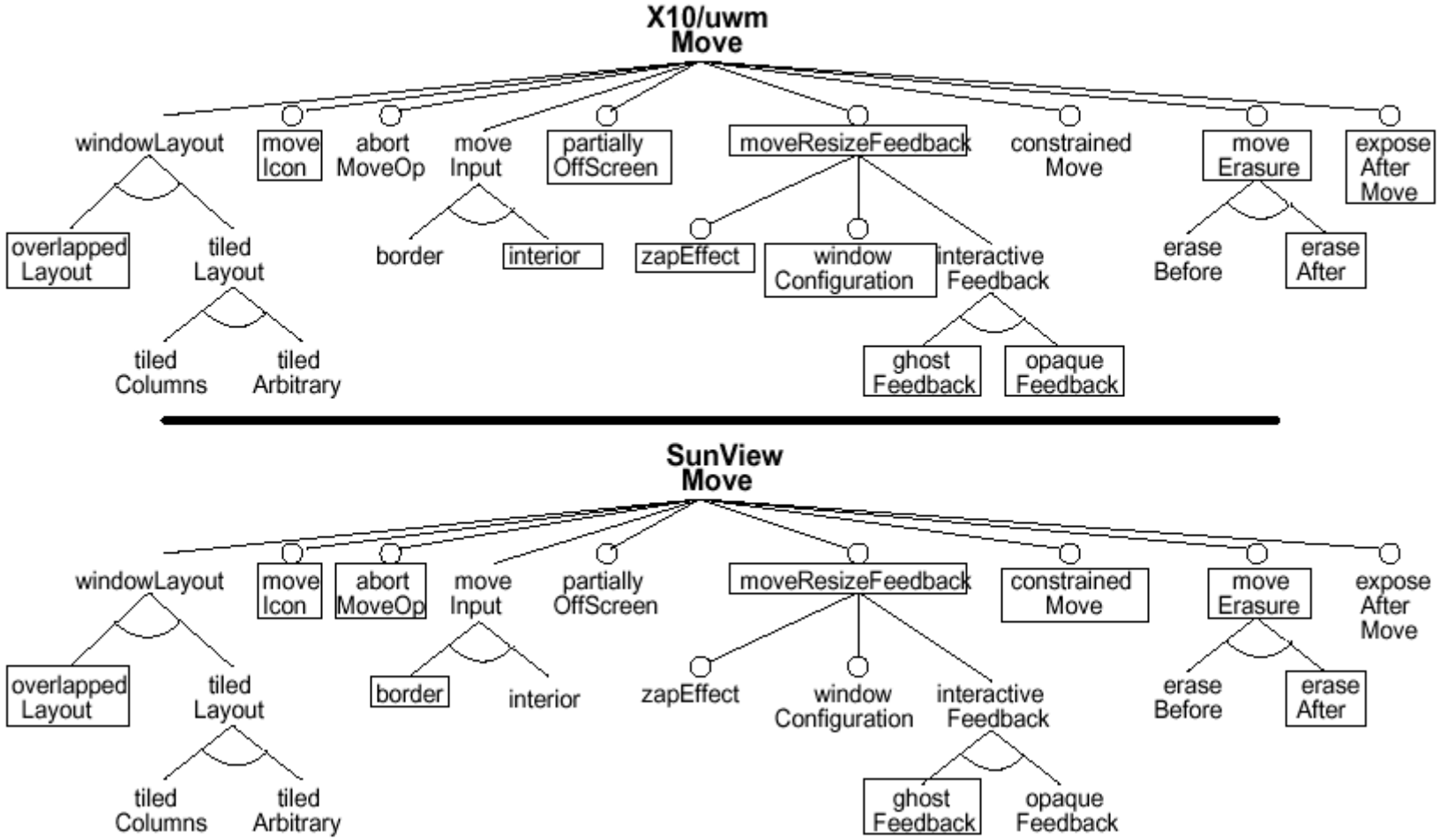
WMS domain model / simple



WMS feature diagram



WMS comparison



<feature1> (requires | mutex-with) <feature2>

Feature composition rules

- `opaqueFeedback` mutex-with `moveErasure`:
  - If the entire window image is moved, then there is nothing left in the previous position to erase.
- `zapEffect` requires `ghostFeedback`:
  - If the entire window image is moved (i.e., `opaqueFeedback`), then there is no window to draw the final zap lines from.
- `zapEffect` requires `eraseAfter`:
  - If the old window image were erased before the *move* operation, then there would be nowhere to draw the final zap lines from.
- `ghostFeedback` requires `moveErasure`:
  - If `ghostFeedback` is used, then the old window image must be erased at some point, either before or after. Thus one of the two alternatives of `moveErasure` must be selected.
- `exposeAfterMove` requires `overlappedLayout`:
  - An *expose* operation can only be done in an *overlapped* system.

WMS system feature catalog

System/Feature	X10/uwm	VMS Windows	SunView	Mac Windows
moveWindow	*Move*	yes	*Move*	yes
constrainMove	no	no	yes	no
moveIcon	yes	yes	yes	no
moveErasure	after	before	after	
exposeAfterMove	yes	yes	no	yes
zapEffectMove	*	no	no	no
interactiveFeedback	*	ghost	ghost	ghost
partiallyOff	yes	yes	no	yes
abortMove	no	no	*Cancel*	no
selectOrder	actionObject	objectAction	objectAction	objectAction

<name> … name used for feature in this system
“yes/no” … feature does/does not exist in this system
<blank> … no information about this feature
“*” … feature is bound at either activation-time or run-time, but not at compile-time
“—” .. feature is inapplicable in this system

Zusammenfassung

- Warum Domain Analysis?
- Inputs/Outputs, Prozesse, Rollen
- Welche Modelle und Beschreibungen?
- Feature-Oriented Domain Analysis als eine Methode