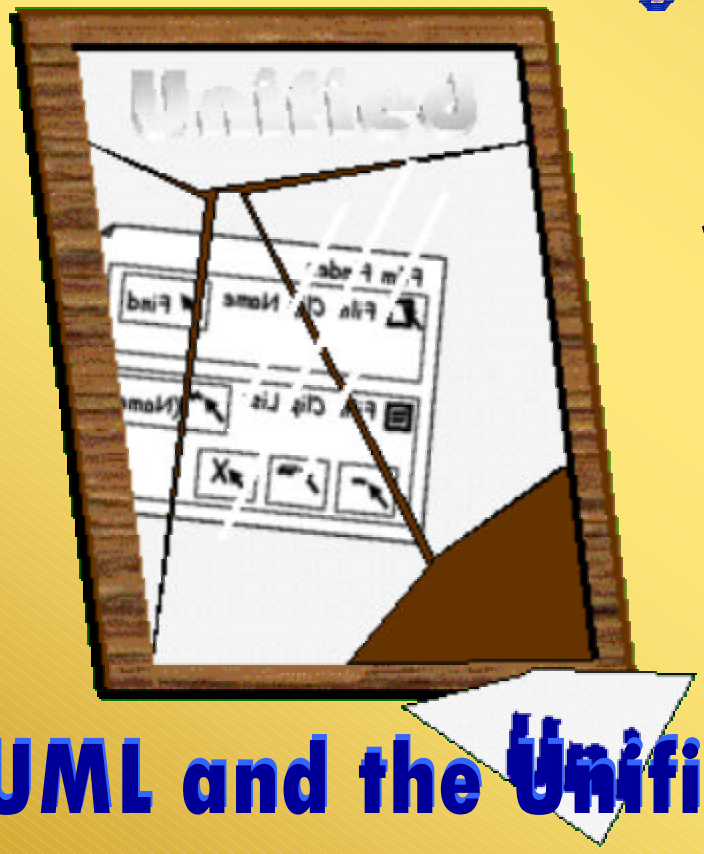
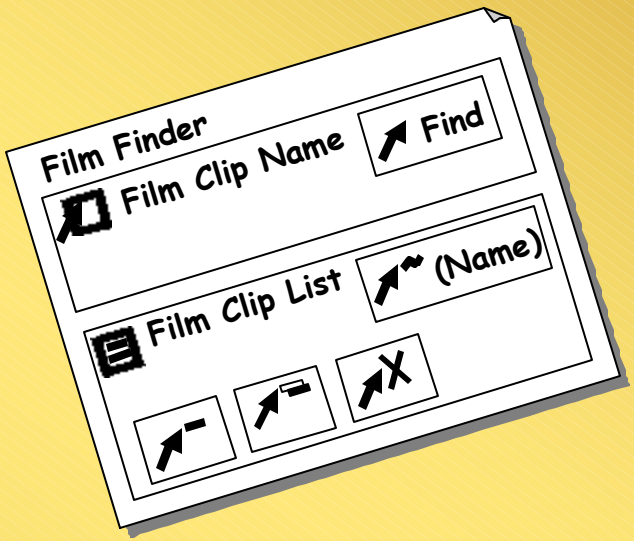
Constantine & Lockwood, Ltd.

TOOLS Pacific 2000 | Sydney | 22 November 2000

The Usability Challenge:





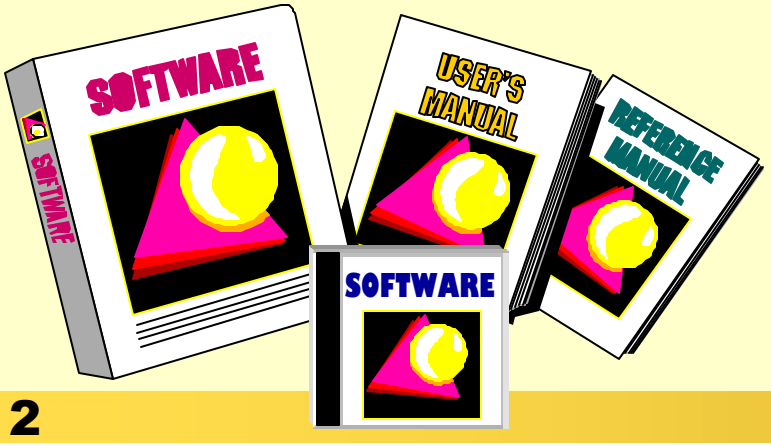
Can UML and the Unified Process Meet It?

Larry Constantine | University of Technology, Sydney

User Interfaces

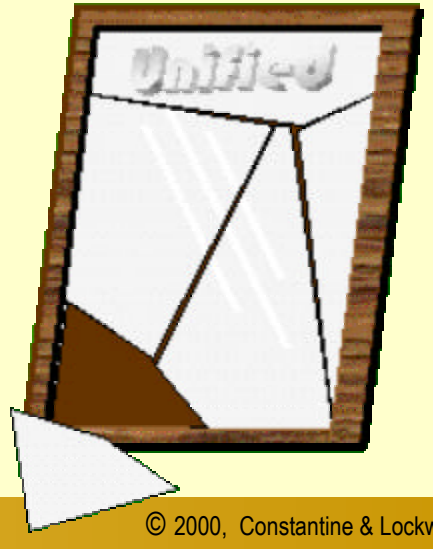
Especially in e-business!

- To users, the user interface IS the system.
- Usability and the quality of the user interface design are often critical factors in product success.
- Increasingly important as a competitive advantage.
- Usability means: ✓ ease of learning and remembering, ✓ efficiency and reliability in use, ✓ satisfaction.
- Usability pays off: ✓ reduced errors, ✓ training, and ✓ support; ✓ increased productivity and ✓ satisfaction.
- To achieve usability: ✓ Understand your users.
  ✓ Understand their work.
  ✓ Model your understanding.
  ✓ Design the UI to fit the models, your users, and their work.



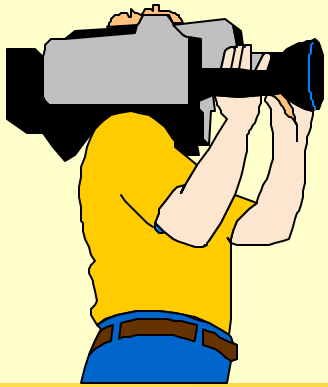
Meeting the Usability Challenge

- Usability in the Process and Language
- Use Cases and Task Models
- Actors and Roles
- User Interface Design



Good News, Bad News

- The Unified Process coupled with the Unified Modeling Language is a single framework comprising a comprehensive and customizable collection of fully integrated concepts, models, notations, tools, tactics, and techniques with supporting training for the disciplined and systematic generation of software solutions in virtually any application in any domain over the full range of scales in system complexity and project time lines.



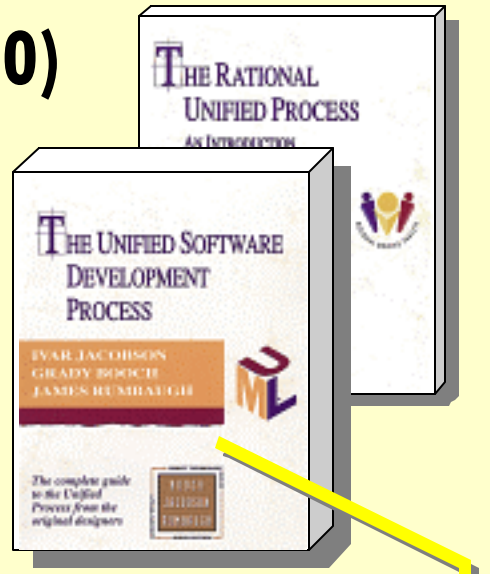
- Almost nobody is actually using it as sold or intended.

Unified Process?

How important is user interface design?

4½ pp. of 262

- **Kruchten,** The Rational Unified Process **(2000)**
- **Jacobson, Booch, Rumbaugh,** The Unified Software Development Process **(1999)**



12 pp. of 450

Sidebar p. 164 on essential use cases!

"We care about user interfaces…only if they are architecturally interesting. However, this is seldom the case."

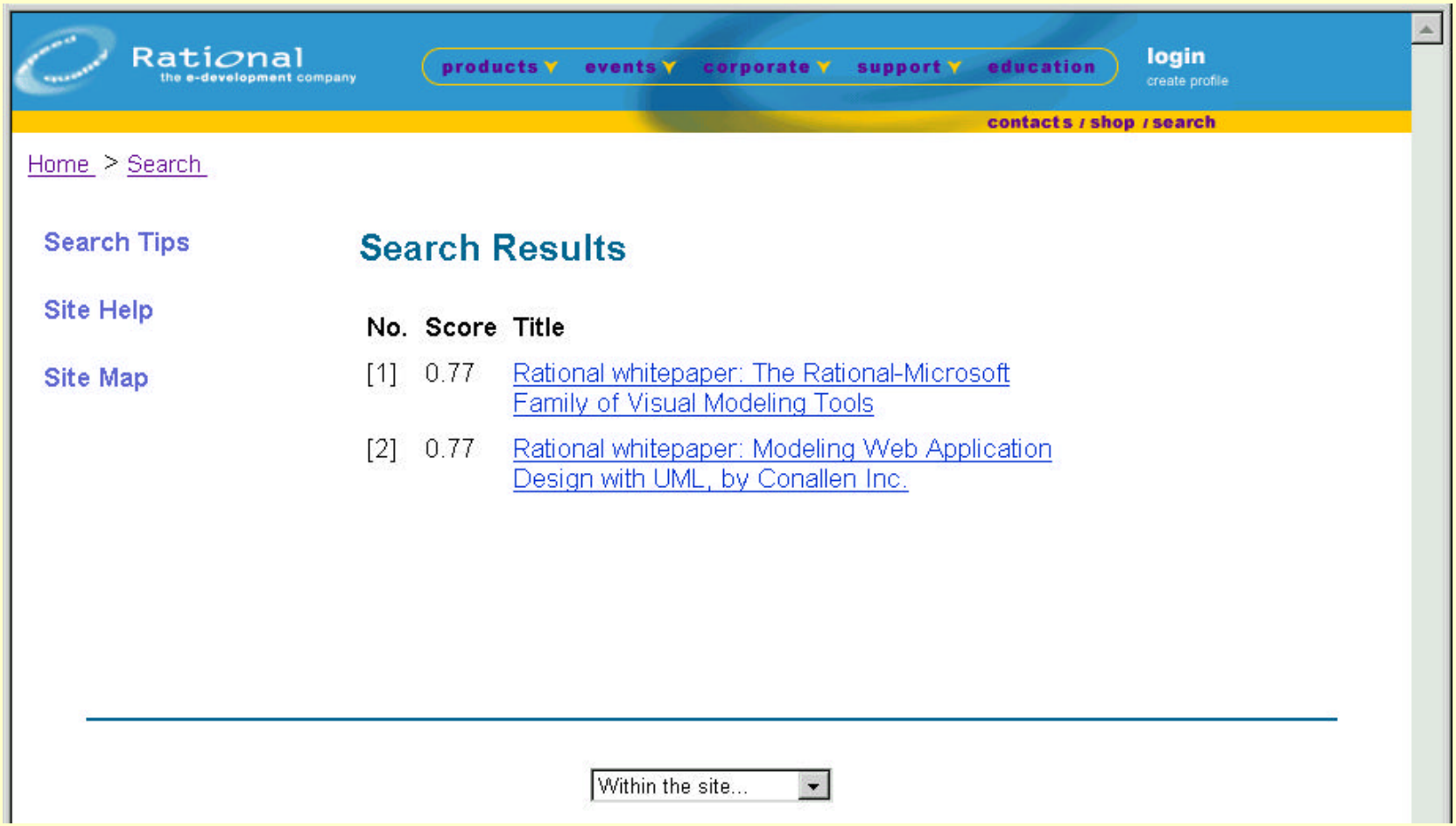
"Based on feedback, we massage the prototype until it satisfies the needs of users….The prototype then becomes the user interface specification."

"Designing the user interface is part of the requirements workflow and not design."

Design is not design?

Unified Process?

But, the Unified Process is a dynamically evolving product. That is, of course, process.
So, what is available today on the official Web site?



Unified? Not!

UML

- ✗ No model or notation for user interface contents.
- ✗ No model or notation for user interface architecture.
- ✗ No complete specification of use case contents.
- ✗ Inadequate narrative form of use case contents.
- ✗ Only covers object-oriented systems.

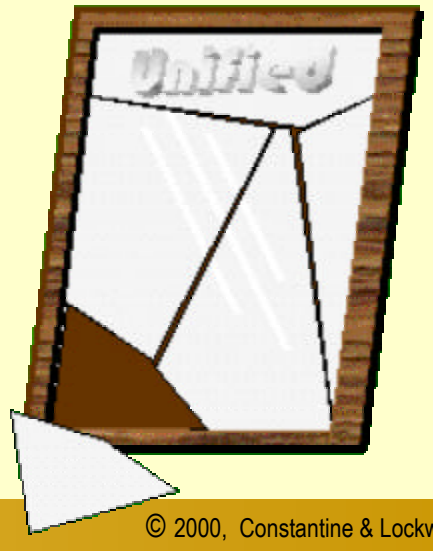
Unified Process

- ✗ User interface design as requirements activity.
- ✗ Simplistic notion of logical and physical prototypes.

- ✗ User interface design by prototyping.
- ✗ Limited concept of user roles.
- ✗ Limited user involvement.

Meeting the Usability Challenge

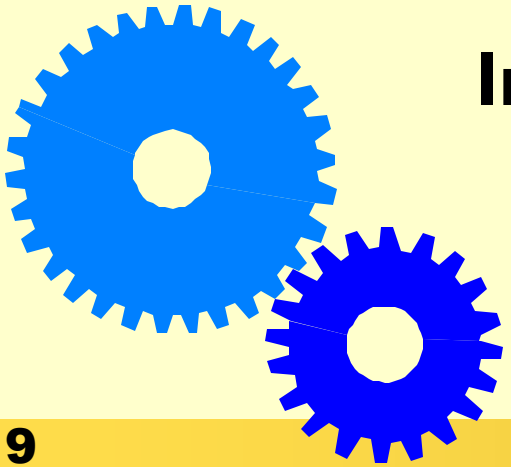
- Usability in the Process and Language
- **Use Cases and Task Models**
- Actors and Roles
- User Interface Design



A Use Case Is

"A coherent unit of functionality **provided by a… system**, subsystem, or class as manifested by sequences of messages **exchanged [between] the system** and one or more outside users (represented as actors), together with actions **performed by the system**."*

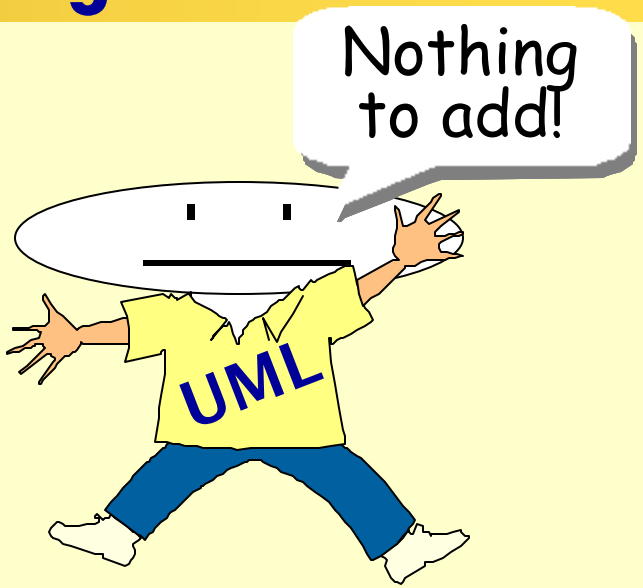
"A description of a set or sequence of actions, including variants, **that a system performs** that yields an observable result of value to a particular actor."**



In UML and UP, use cases are about the system and the interaction from the system point of view!

*Rumbaugh et al., 1999 **Jacobson et al., 1999

The Language of Use Cases



Structure and contents

- What exactly do use cases contain?
- How are the contents organized?
- What do the contents mean?

Idiomatic expression

- What language and forms are used to express the contents that define use cases?
- How are these understood by software that supports use cases in analysis and design?

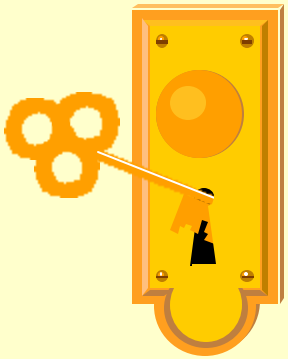
Interrelationships

- How are use cases interrelated and how are these connections expressed within the content of use cases?

From Use Cases to User Interfaces

- **Usage-centered design** was motivated by limitations of usability testing and other human-factors approaches.
- Many thought use cases were natural for user interface design, yet early results were disappointing.

Key to success is a shift of focus from interaction to INTENTION, from elaboration to SIMPLIFICATION.



- It has ben applied & taught since 1993, published 1994, 1995; complete text published 1999.
- Others have since recognized need to revise and extend use cases by abstraction and focus on purpose or goals:
  - ✓ Kaindl (Austria, 1995)
  - ✓ Graham (U.K., 1996)
  - ✓ Cockburn (U.S., 1997)
  - ✓ Lee & Xue (China, 1999)

Use Case Contents

- **Name** - essential purpose (transitive present participle)
- **Identification** - for design process, tools, and tracking
- **Contextual purpose** - meaning and purpose of use case in the larger context of the application and user's work
- **Supported roles** - user roles supported by task case
- **Priority** - (optional) frequency, importance
- **Related use cases** - specializations, etc.
- **Process** -
  - **Preconditions**
  - **External (asynchronous) extensions**
  - **User Intentions** and **System Responsibilities** } separated, numbered steps
  - **Internal extensions** (optional)
  - **Post-conditions**
- **Business rules** - attached or linked where and as needed

Structured Task Case Example

Simplified but structured narrative (user-friendly form):

running predefined test

Purpose: setup and conduct standard test with or w/no

Supports: routine test operator

Preconditions: system initialized, authorized operator logged on

USER INTENTIONS	SYSTEM RESPONSIBILITIES
optionally at any point do: canceling test; or resetting entry	**optionally at any point do:** reporting error
2. Pick desired test	1. Show list of standard tests 3. Show chosen test setup
4. **in any order:** { **optionally:** set Print option; **optionally do:** adjusting test setup } 5. Start test	6. Run test 7. **do:** reporting test results

Postconditions: machine stopped

may include explicit or implicit task cases

optional content

unordered actions

optional actions

uses this task case

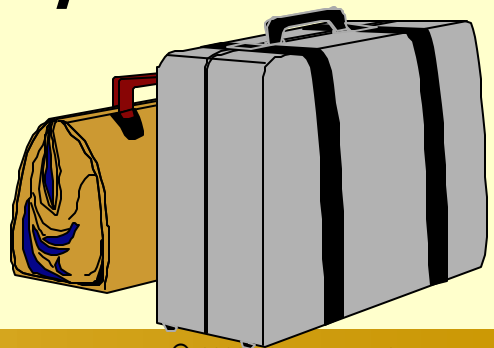
extensions (events may occur anytime)

Related Cases

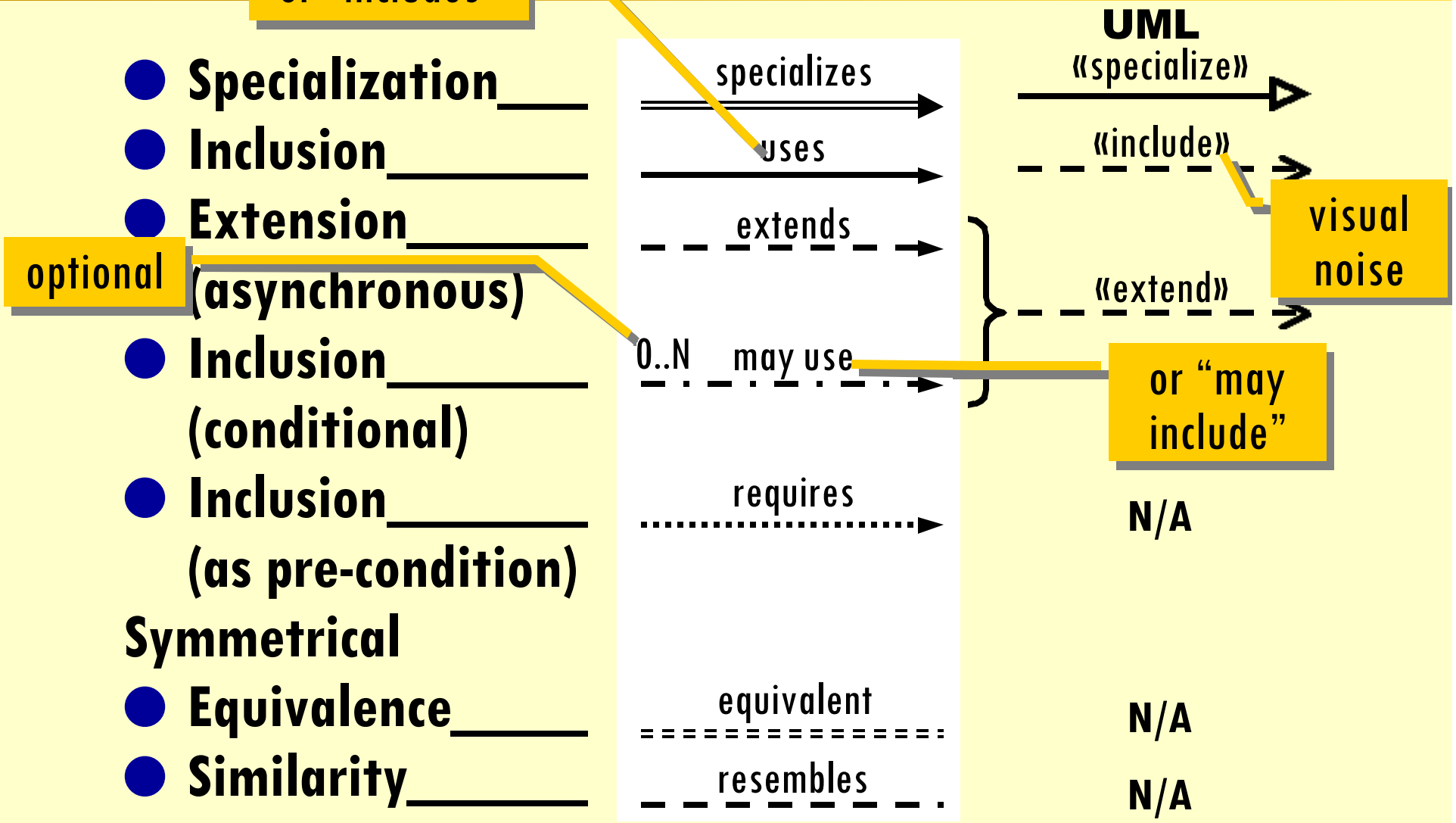
Additional and more refined relationships needed:

- Affinity - generic and unspecified similarity
- Specialization - a use case may be derived as a more specialized version of another use case.
- Inclusion - a use case may include or use other use cases as a part of its process.
- Conditional inclusion - a use case may optionally include or use another use case at some specific point in its process.
- Asynchronous extension - a use case may be altered or extended by use cases that could occur at any point in its process.

Both are "extensions" in UML.



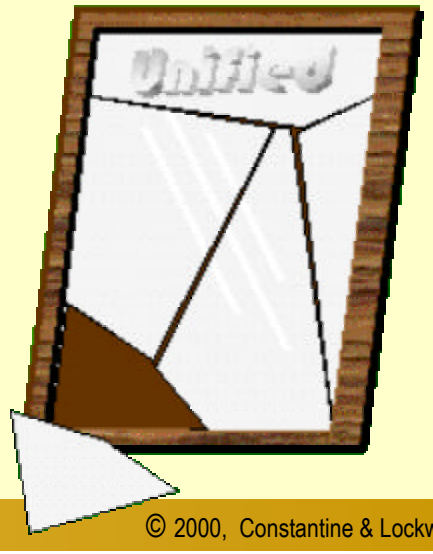
Summary of Notation



Distinct line styles and clearer, specific labels enhance readability, especially in large, complex models.

Meeting the Usability Challenge

- Usability in the Process and Language
- Use Cases and Task Models
- **Actors and Roles**
- **User Interface Design**



Important Roles

- Use cases are enacted (performed) by users in some role or relationship with a system.
- The role or relationship, not the actor or user who occupies the role, is important for interface and interaction design.

In UML: "An actor is a role that a user plays with respect to the system."*

- We call them roles because they are!
  Facilitates communication with users.
  Avoids confusing people with roles they play.

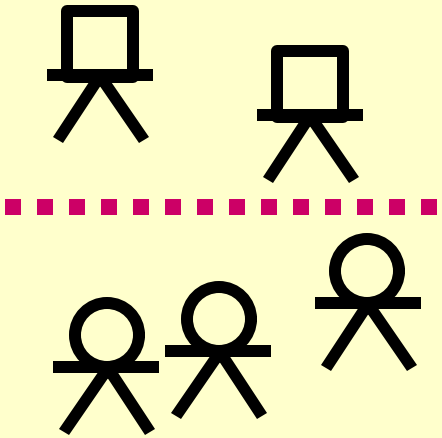
  Aids thinking abstract relationship, not concrete person.

*Fowler, 1998

Actors' Equity

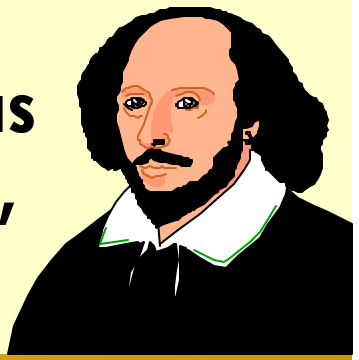
UML and UP focus on actors, not users. So what's wrong?

- **Concept of actors not well-defined or elaborated.**
- **Two kinds are lumped together.**



  - **System actors: other systems (software or hardware) with which system interacts.**
  - **Human actors: users taking on some role in relation to the new system.**
  - **For usability and user interface design, only user roles are directly relevant.**

- **Roles are abstractions: characteristic needs, interests, behaviors, expectations, and responsibilities.***
- **User roles can be elaborated as profiles, such as incumbent characteristics, interaction patterns, information flow, usability criteria, etc.**



Mapping User Roles

User roles may be related in numerous ways -

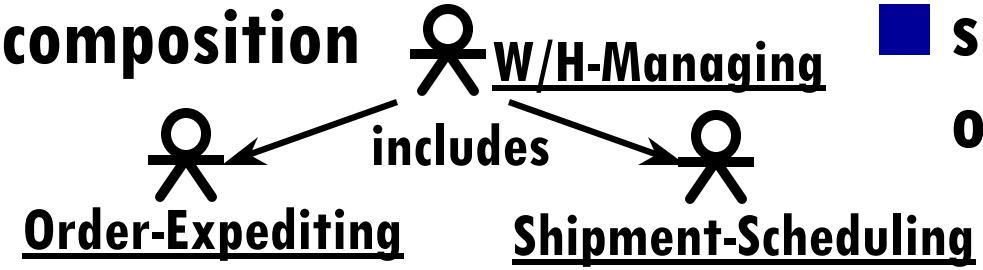
affinity

- unspecified similarities or common characteristics

classification

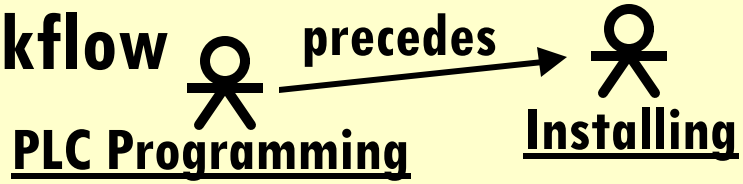
- specializations of another more general role

composition



- some may combine behaviors or characteristics of others

workflow

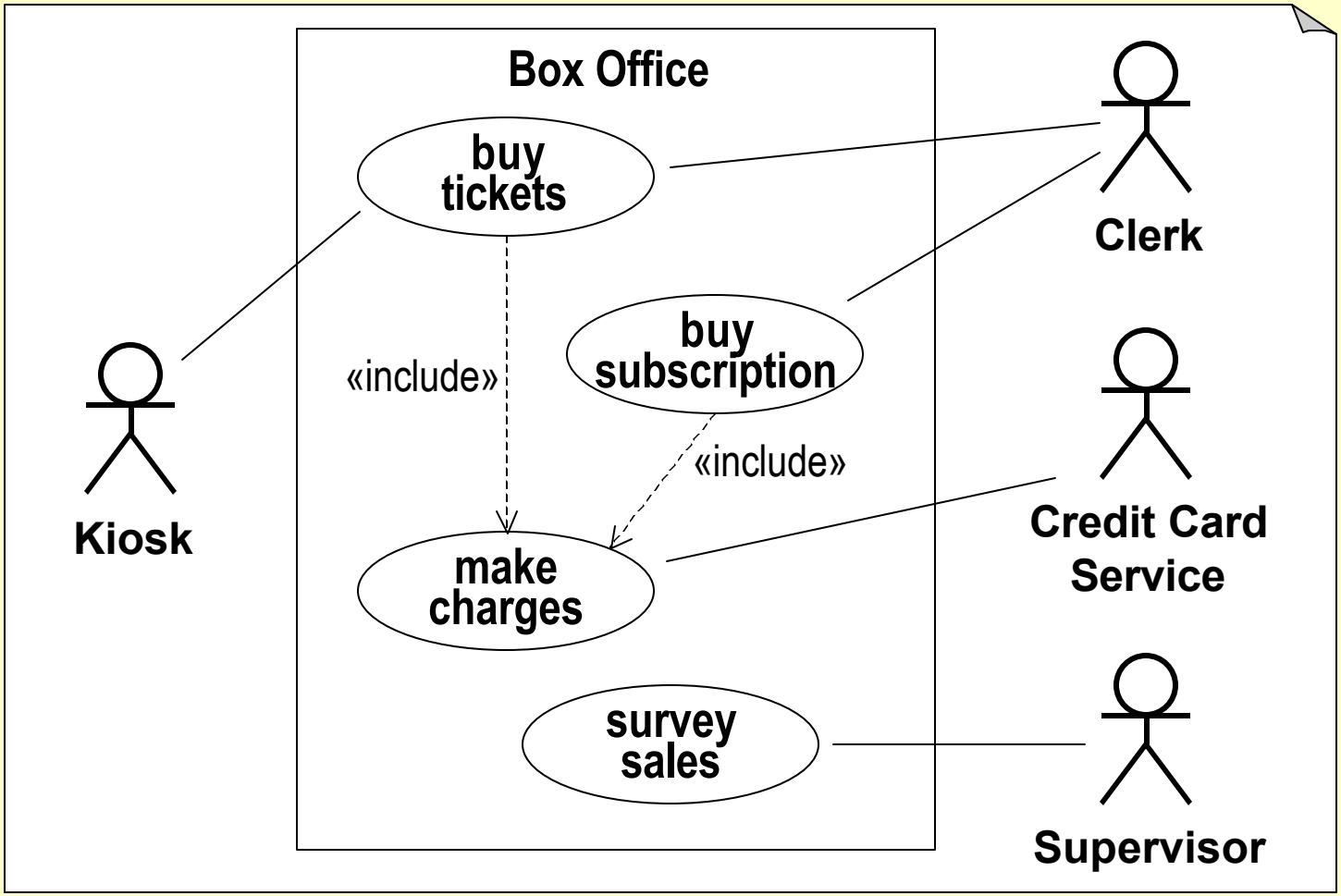


- sequential process, supplying or using information, etc.

Other relationships as needed for problem, e.g., equivalence.

Use Case Diagrams

- A typical textbook problem: 4 use cases, 4 actors.*

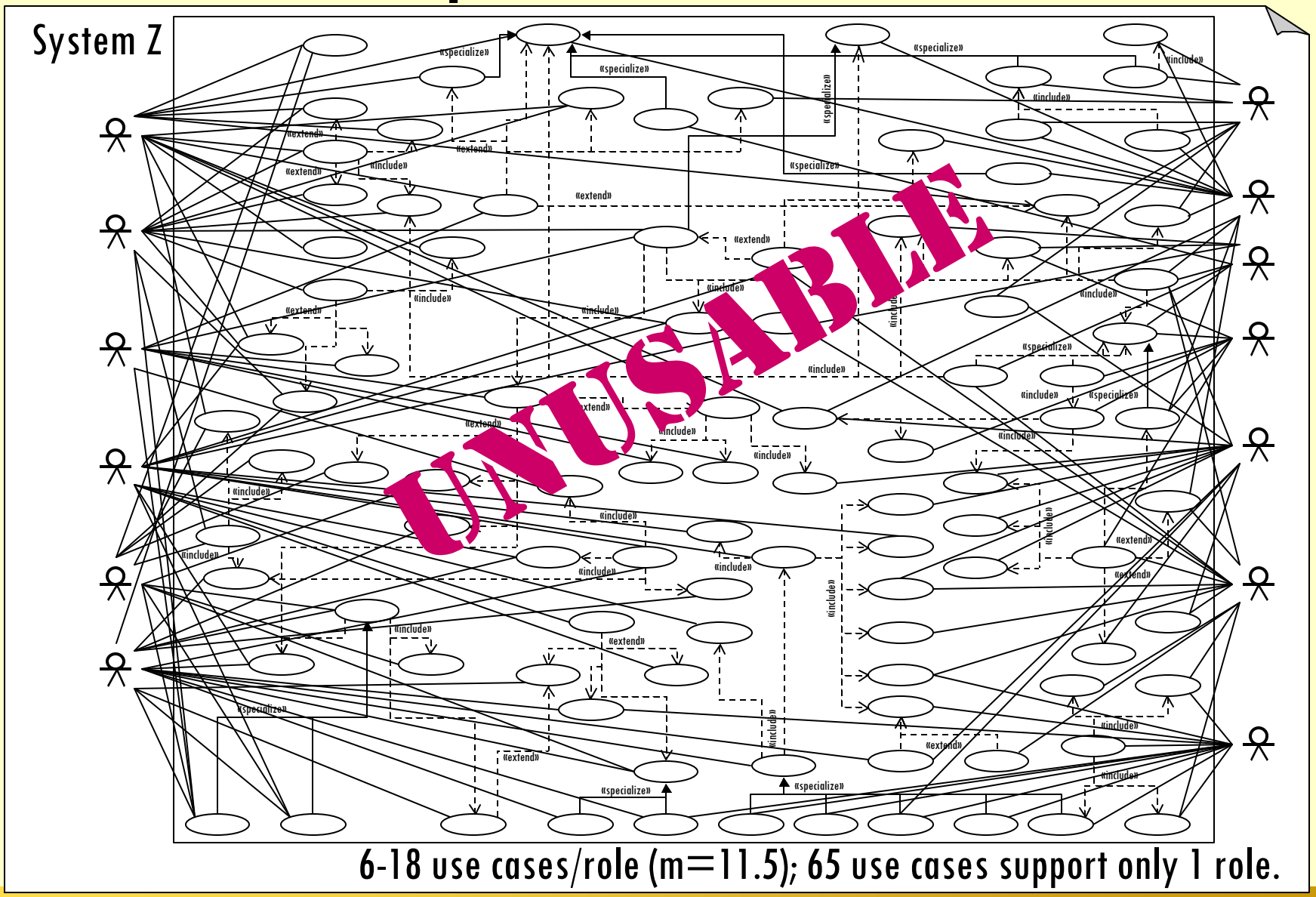


- Simple, straightforward, understandable model?

* Rumbaugh et al., 1998

Use Case Diagrams

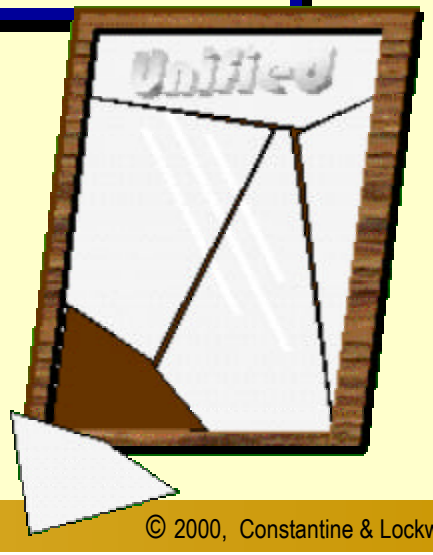
- **Modest real-world problem: 100 use cases, 13 user roles.**



6-18 use cases/role (m=11.5); 65 use cases support only 1 role.

Meeting the Usability Challenge

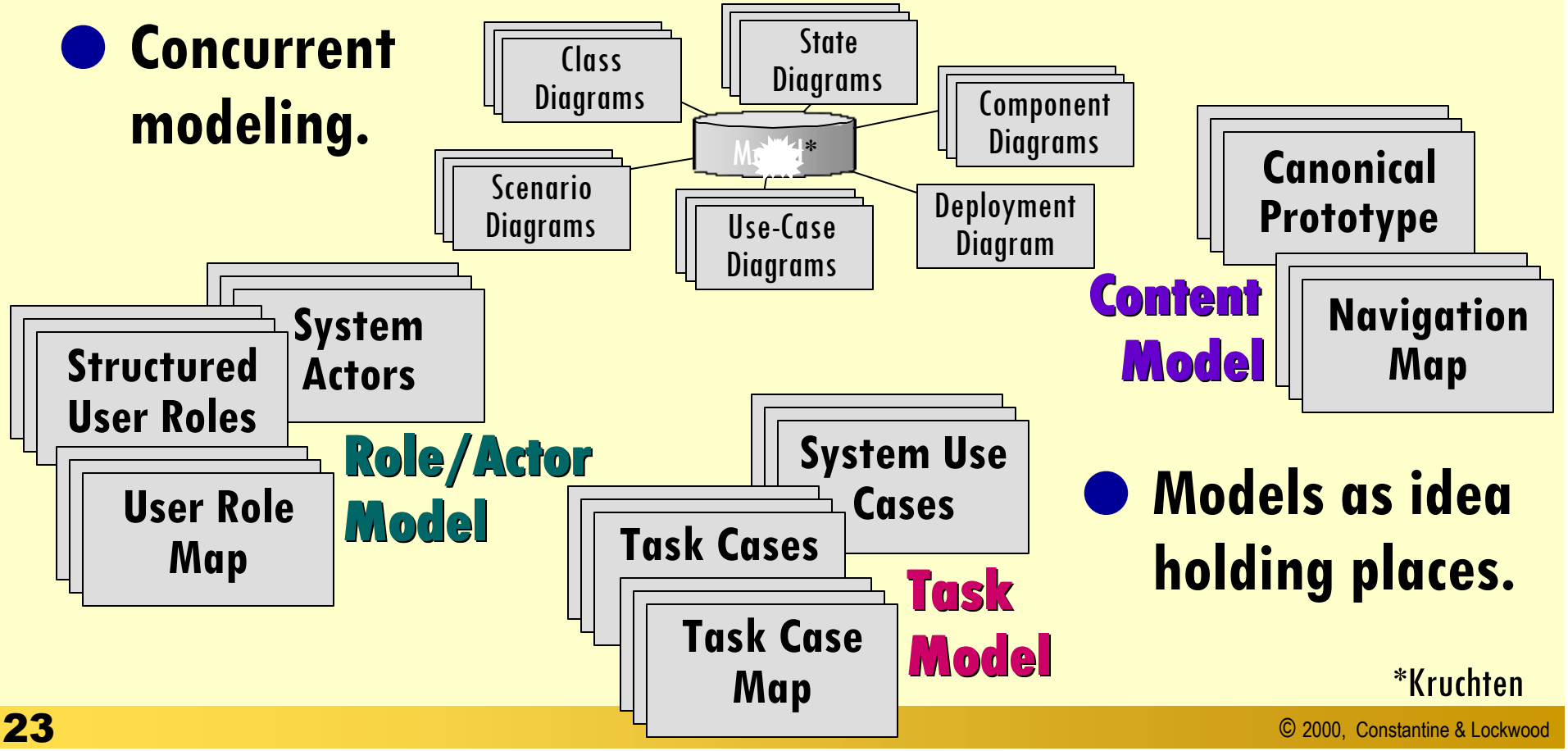
- Usability in the Process and Language
- Use Cases and Task Models
- Actors and Roles
- **User Interface Design**



From Unified to Usable

What is needed for a truly usage-centered process?

- **Some models can be fudged in UML, others cannot or can only be awkwardly approximated.**
- **Improve workflow: UI design as model-driven design.**
- **Concurrent modeling.**



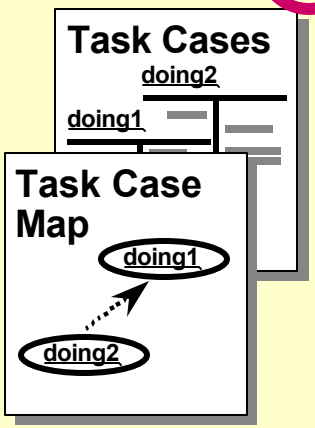
- **Models as idea holding places.**

*Kruchten

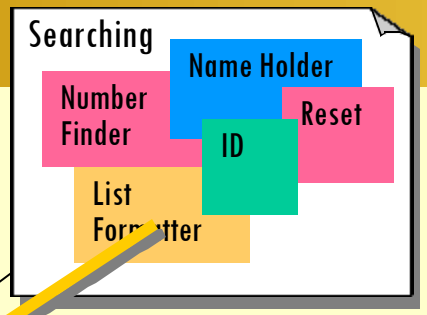
Bridging the Gap

Well-conceived models are needed to bridge the gap from use cases to visual and interaction design.

abstraction



TASK MODEL

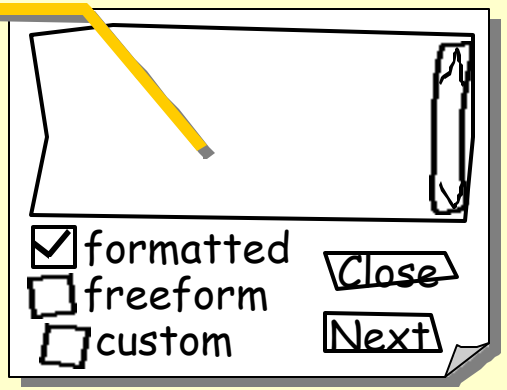


CONTENT MODEL

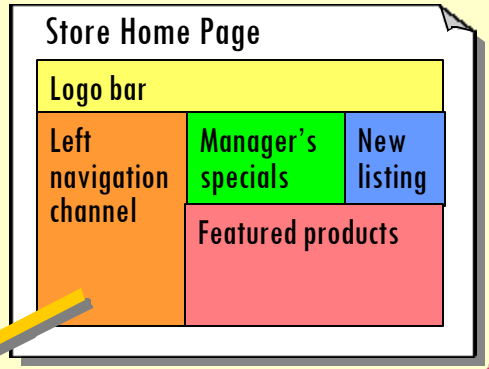
Abstract contents only

Rough but representative

LOW-FIDELITY PROTOTYPE



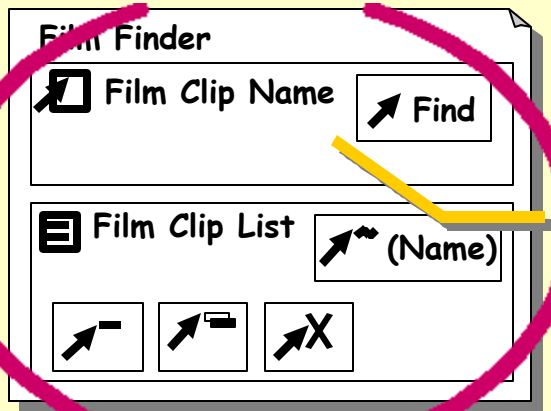
Now Easy steps! Now magic happens!



WIRE-FRAME SCHEMATIC

Layout (size, position) and importance

New!

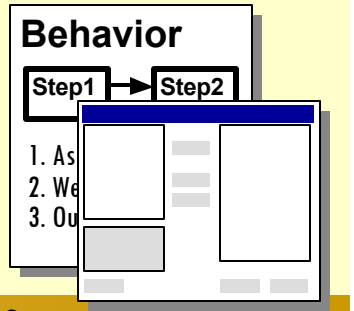


CANONICAL PROTOTYPE

Standardized abstract components

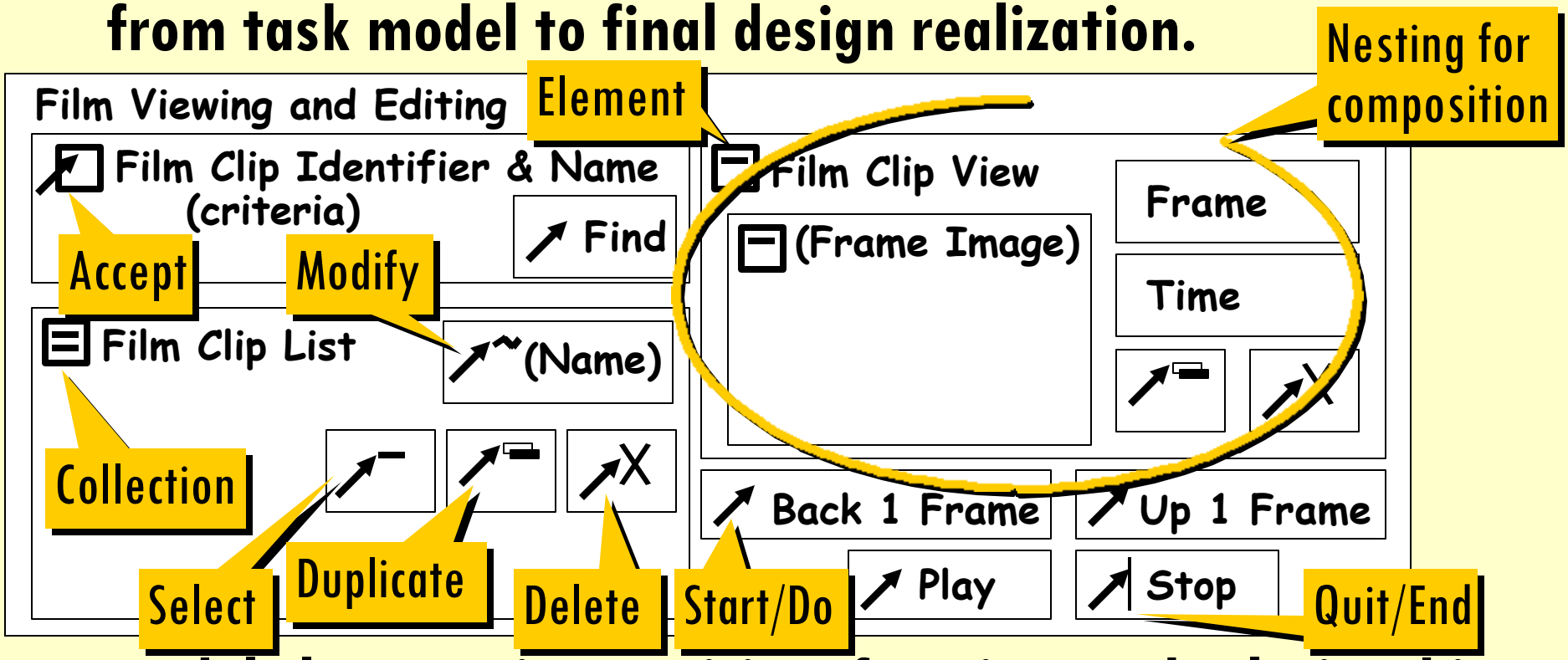
realization

DESIGN



Canonical Prototypes

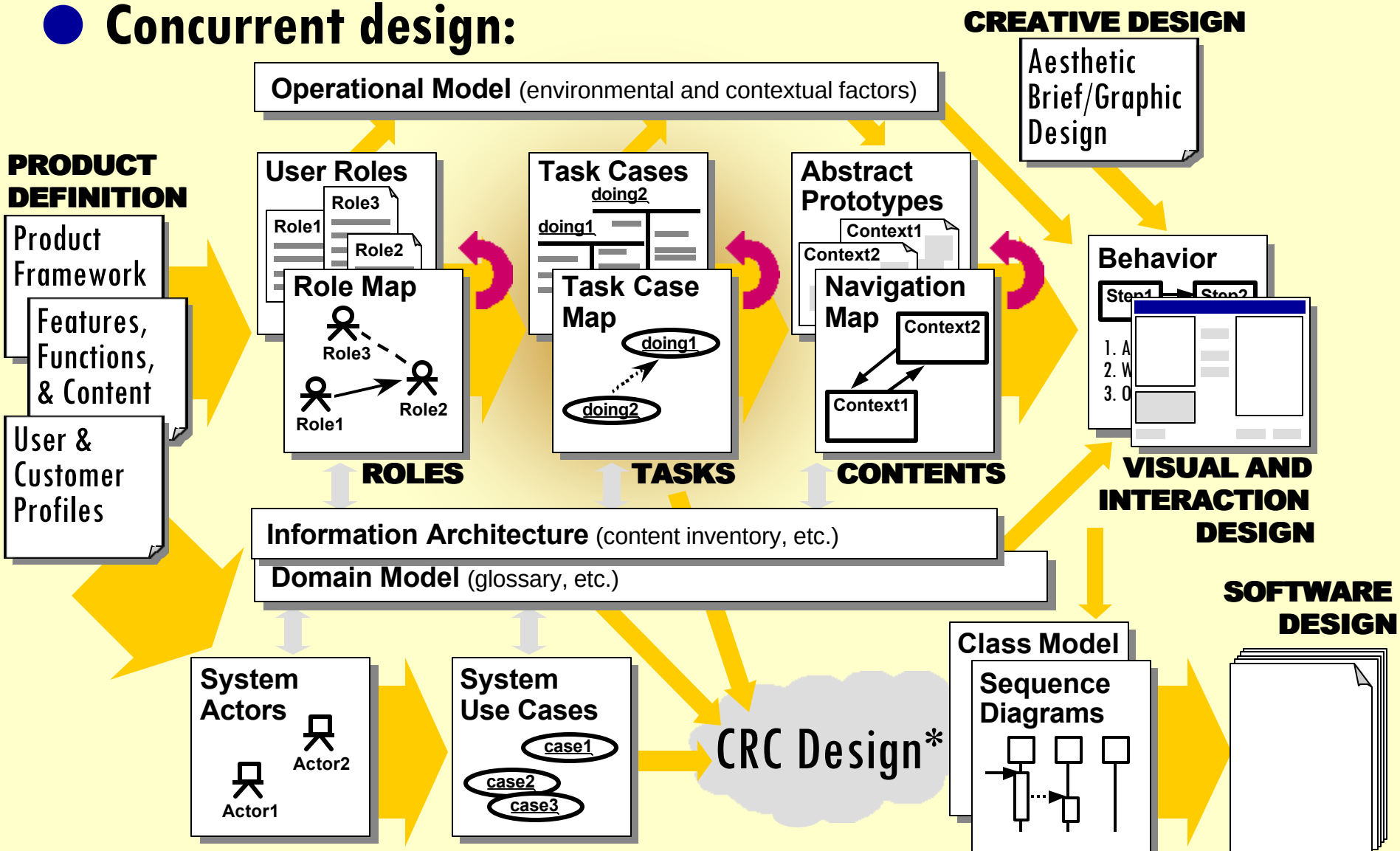
- Abstract prototypes constructed from standardized set of abstract components with specific functions.
- Intermediate level of abstraction smoothes transition from task model to final design realization.



- Models layout, size, position, function, and relationships of interface components, but NOT detailed appearance.

Usage-Centered Design Process

- **Concurrent design:**



- **UI and software logical architecture!**

* Noble, after Wirfs-Brock

From Unified to Usage-Centered

Fix the UML notation:

- Distinguish system actors and user roles.
- Support structured contents of roles and use cases.
- Improve "use case diagram," especially interconnections.
- Add models and notations for abstract prototypes and interface navigation maps.

Fix the Process:

- Expand user involvement in key activities.
- Concurrent modeling process for system actors and use cases and for user roles and task cases.
- Model-driven user interface design as a design process using abstract prototypes and content navigation maps.
- Responsibility-driven design process of software internals.

Philosophical Differences

- One size does **not** fit all, especially if it is the extra jumbo deluxe heavyweight all-in-one unified edition.
- A simple process that can be scaled up is far better than a complex process that can be cut down.
- Every diagram, document, activity, or artifact must save time, improve quality, or facilitate problem solving.

Products not process! Consumables not deliverables!

- Model only what helps and only to the extent and precision that helps.
- Leave rigor and obsessive completeness and correctness to the academics. Er, um.
- To users the user interface IS the system. UI design is not an afterthought.