Analysis

• Study of a system (e.g., a software system to be built)

• Objectives
  – understand system
  – document understanding
  – promote a common understanding among participants (e.g., clients, analysts, developers, users, managers)

• Key activities
  – studying
  – learning
  – writing
  – communicating

Analysis of Customer Requirements

There’s more than just systems analysis.

• Understand application context
  – general business objectives
  – specific business needs to be addressed by the system

• Understand business constraints
  – cost
  – schedules
  – legal considerations

• Understand the technical constraints
  – available technical resources (hardware, software, personnel)
  – technology and expertise that can be brought in for the project
Chapter 7 - Needs and Feasibility Assessment

Using OSM for Needs and Feasibility Modeling

- Standard guide to assessing needs and feasibility
- Standard, but can be tailored for particular projects
- Can be prepopulated with data and used for several projects
- Same model for assessment and application development
- Can query the assessment model with standard query languages (in the same way as the application model)
Query Languages for OSM

- OSM-QL
- OSM-Algebra
- OSM-Calculus
- OSM-SQL

OSM-QL

List goals for the primary alternative.

List suggested alternatives that satisfy all needs.
OSM-Algebra

List goals for the primary alternative.

\[ \pi_{\text{Goal}}(\text{Goal} \ | \ \rho_{\text{Need - Goal}}^{\text{Suggested Alternative satisfies Need}} \ | \ \rho_{\text{Primary Alternative - Suggested Alternative}}^{\text{Primary Alternative}}) \]

List suggested alternatives that satisfy all needs.

\[ \text{Suggested Alternative - } \pi_{\text{Suggested Alternative}}(\text{Suggested Alternative | Need - satisfies}) \]

OSM-Calculus

List goals for the primary alternative.

\[ \{ <x> | \exists y(\text{Goal}(x) \land \text{Suggested Alternative}(y) \text{satisfies Need}(x) \land \text{Primary Alternative}(y)) \} \]

List suggested alternatives that satisfy all needs.

\[ \{ <x> | \text{Suggested Alternative}(x) \land \forall y(\text{Need}(y) \land \text{Suggested Alternative}(x) \text{satisfies Need}(y)) \} \]

\[ \{ <x> | \text{Suggested Alternative}(x) \land \forall y(\lnot\text{Need}(y) \lor \text{Suggested Alternative}(x) \text{satisfies Need}(y)) \} \]

\[ \{ <x> | \text{Suggested Alternative}(x) \land \lnot\exists y(\text{Need}(y) \land \lnot\text{Suggested Alternative}(x) \text{satisfies Need}(y)) \} \]
**OSM-SQL**

List goals for the primary alternative.

```
select Goal
from satisfies, Goal, Primary Alternative
where Suggested Alternative = Primary Alternative and Goal = Need
```

List suggested alternatives that satisfy all needs.

```
select Suggested Alternative
from Suggested Alternative A
where not exists (select *
    from Need N
    where not exists (select *
        from satisfies S
        where N.Need = S.Need and
        A.Suggested Alternative = S.Suggested Alternative))
```

---

**Analysis Methods**

- Analysis is a knowledge-discovery task
  - learning is never easy
  - requires deep involvement with information
  - often faced with vague and contradictory information
  - experience counts

- **General approaches**
  - ask questions, observe activities, read documents and forms
  - learn by formal writing and by model construction

- **Specific approaches**
  - methodologists: e.g., scenarios, use cases, CRC cards, …
  - proprietary approaches often used in consulting firms
  - model-driven approach (OSM)
OSM Model-Driven Analysis

• Ontological model
  – models can either expand or limit our ability to analyze
  – best: models that let us directly record structural and behavioral concepts as found in the system being modeled

• Match analysis approach with application characteristics
  – ORM approach
    • data intensive application components
    • examples: personnel records, business data, scientific data
  – OBM approach
    • components with intensive individual object behavior
    • examples: behavior drivers for elevators, machining, traffic lights
  – OIM approach
    • components with intensive interactive behavior
    • examples: automatic-teller machines, airline reservations, sensors

Application-Model Integration

• Large projects
  – several analysts each tackle a manageable-size component
  – systematically integrate these components

• Approach to integration
  – integration framework and strategy
    • framework: high-level OSM application model
    • strategy: decisions about how to proceed and how to resolve conflicts
  – integration activities
    • compare: identify correspondences and conflicts
    • conform: resolve conflicts
    • merge: put components together and reorganize as warranted
Example: Integration Framework

High-level diagrams work well for specifying integration frameworks.

Example: ORM Integration

Name conflict: Name of Room vs. Name of Guest; Person vs. Guest
Structural conflict: reservations
Constraint conflict: minimum participation of Room in reservations
Example: Integrated ORM

![Diagram of a model showing relationships between Room, Guest, and their properties.]

Example: OBM Integration

- Reservation Clerk
  - @ request guest list
    - list arriving guests
  - @ special guest arriving and not yet notified proprietor
    - notify proprietor

- @ cancel reservation

- @ new reservation
  - make reservation

- Ready

- Exists

Chapter 7 - 15

Chapter 7 - 16
Example: Integrated OBM

Reservation Clerk

- new reservation
  make reservation

- request guest list
  list arriving guests

- special guest arriving
  and not yet notified proprietor
  notify proprietor

- cancel
  cancel reservation

Analysis Validation

- Checking a model for completeness and correctness
  - Discovery (Chapter 7)
  - Formalization (Chapter 8)
  - Requires skill and insight ("Ya gotta use your head."")

- Discovery techniques
  - Formal technical review
    - systematic scrutiny based on queries, behavior scenarios, and interaction sequences
    - white-box and black-box “testing”
  - Object classification – cohesive sets of objects
    - structural similarity
    - behavioral similarity
White-Box Validation

- “Fire” each transition
- “Execute” each state net
- Check correctness
- Check completeness
  - Are all expected/required scenarios addressed?
  - Are all exceptions handled?

Black-Box Validation

- ORM perspective
  - Pose and “execute” ad hoc queries.
  - Does the application model contain the information necessary to answer these queries?
- OIM perspective
  - Consider stand-alone interactions and interaction sequences.
  - Does the application model include all interactions and interaction sequences required in the system being built?
- OBM perspective
  - Check the receiving end of interactions.
  - Are all required application-model interactions handled?
Object Classification

• Why classify?
  – may discover new object sets
  – may cause us to better organize object structure and behavior
  – may help us better understand an analysis application model
• Classify based on similarity
  – the more two objects are related to other objects in the same way, the more they are similar
  – the more two objects behave the same, the more they are similar

Congruency

• An object set is congruent if the common properties of the objects in an object set $S$ coincide with the properties explicitly defined for $S$.
• Explicit properties – given in application model
• Common properties
  – Common relationship-set property: all objects participate
  – Common state-net property: all objects can be in state or transition
  – Common interaction property: all objects can interact the same
• Incongruent:
  – Overstatement: explicit property not common to all
  – Understatement: common to all but not explicit
Structural Congruency

Congruent:

\[
\begin{array}{c}
\text{Room} & \overset{1}{\text{has}} & \overset{1}{\text{RoomNr}} \\
\end{array}
\]

Incongruent:

\[
\begin{array}{c}
\text{Room} & \overset{0:1}{\text{has}} & \overset{1}{\text{Occupies}} \\
\text{Guest} & \overset{1}{\text{has}} & \overset{1}{\text{Room Nr}} \\
\end{array}
\]

Congruent:

\[
\begin{array}{c}
\text{Room} & \overset{1}{\text{has}} & \overset{1}{\text{RoomNr}} \\
\text{OccupiedRoom} & \overset{1}{\text{has}} & \overset{1}{\text{Guest}} \\
\end{array}
\]

Behavioral Congruency

Incongruent

Checkout Clerk:

\[
\begin{array}{c}
\text{Checkout Clerk: Checking Bill} & \text{OK} & \text{pay bill} \\
\text{Checkout Clerk: Checking Bill} & \text{Not OK} & \text{register complaint} \\
\text{Checkout Clerk: Checking Bill} & \text{display bill} \\
\end{array}
\]

Congruent

Checkout Clerk:

\[
\begin{array}{c}
\text{Checkout Clerk: Checking Bill} & \text{OK} & \text{pay bill} \\
\text{Checkout Clerk: Checking Bill} & \text{display bill} \\
\text{Checkout Clerk: Checking Bill} & \text{register complaint} \\
\end{array}
\]

Chapter 7 - 23

Chapter 7 - 24
Congruency and Integration

Integrated But Not Congruent

Missing Generalization (Room)

Missing Specializations (CurrentGuest, FutureGuest)
Overstatement
Explicitly Defined but not Common

Overstatement Resolution
Add Missing Specializations
**Understatement**

Common but not Explicit

![Diagram showing Understatement](image)

**Understatement Resolution**

Move Explicit Property to Generalization

![Diagram showing Understatement Resolution](image)
Integrated and Congruent

Room

1:*

has

1:*

View

OccupiedRoom

1

Name

Guest

has

1

ReservedRoom

1:*

has reservation for

CurrentGuest

1:*

FutureGuest

has

1

occupies

1:*