Roadmap UI Design Workshop

(A workshop to develop the Conceptual Design of a GUI or WebUI based product)

Prepared by:

Jim Berney

The Usability Group, LLC.

jim.berney@usability.com

www.usability.com

Nielsen RCM Roadmap & Bridge Workshop

- Introductions

- Workshop goals
  - Identify roles and responsibilities of the design team. What are the expected interactions between team members?
  - Develop a product design statement - agree on scope
  - Develop a list of activities and milestones for RCM - identify responsibilities, constraints, dependencies, timeframes
  - Develop a User Profile and identify Usability Objectives
  - Develop a set of task flows and a task list for each User Profile
  - Develop a set of user objects
  - Build a paper prototype of the RCM Conceptual Model using the MyEureka! Tool as the UI framework.

- Agenda

Identify Project Stakeholders

Name	Company	Role	Responsibilities

Identify Stakeholder Issues

Issue	Owner

Identify Project Constraints

Constraint	Impact on Design

Develop a Product Concept Statement

- *Goal*:
  - Develop a Product Concept statement that supports both Product Objectives and Marketing Objectives.
    - Take the user's perspective
    - Each user profile may have a different role within the product, but each role must support the "Product Concept".
    - All project stakeholders should agree on the scope of the project and boundaries for design. Agree on what the product is and is not.

- *Output*:
  - A one sentence Product Concept statement

- *Exercise*:
  - What is the product name?
  - Brainstorm the general goals and functionality of the product.
  - List the user profile categories identified for the product.
  - Complete the following statement:
    - [Product Name] is a [noun that describes product] that performs [what] for [user profile names].

What is User-Centered Design (UCD)?

- User Centered Design (UCD) is a set of analysis, design, and testing activities that focus on the user.

- UCD activities involve users and project stakeholders early and throughout the product lifecycle (analysis, design, construction, and testing).

- UCD activities that will be performed during this workshop include:
  - Define and integrate User Profile
  - Define Usability Objectives and integrate into product requirements
  - Perform Task analysis
  - Perform Usability Testing (informal testing will be performed throughout this workshop
  - Build a paper prototype to facilitate iterative testing and design
  - Develop a UI Specification of the resulting design

What is User-Centered Design (UCD)?

- This workshop is a Participatory Design activity that involves a combination of users and project stakeholders:
  - Users (actual users from each “user profile” should be represented)
  - Product Manager
  - Subject matter experts (domain experts)
  - System engineers (represent the Database and backend process design)
  - Engineers/Developers (responsible for the UI)
  - Technical Communicators (responsible for developing training, project documentation, tutorial, on-line help, etc.)
  - Usability engineer (assigned to project)
  - Manager(s) of end-user groups (internal applications only)

The Usability Roadmap

- The Usability Roadmap represents the 10 most important steps towards designing a usable and successful project. We believe that the Roadmap is the foundation for a successful design.

- Below are the Ten Steps of the Usability Roadmap.
  - Define Usability Objectives
  - Perform User Analysis and build a User Profile
  - Perform Task Analysis
  - Define roles and responsibilities for each member of the Design Team
  - Implement a UI Design Process
  - Develop a Usability Evaluation Plan
  - Develop a Project Style Guide
  - Document the UI Specification and build Requirements
  - Develop a strategy for User Assistance
  - Maintain a Feedback Channel from the user community

- During this workshop, we will initiate the following Roadmap activities:
  - Define and integrate User Profiles
  - Integrate Usability Objectives into the conceptual design
  - Perform a high-level Task Analysis for the key User Profiles
  - Implement activities within the UI Design Process (conceptual design, low-fidelity prototyping)
  - Perform informal Usability Testing
  - Develop a draft of the UI Specification
  - Develop a draft of the UI Style Guide

The Yardstick for Design

- The Yardstick represents 10 key components of UI Design. Over the years we have found that all usability problems may be categorized using the Yardstick components.

- Below are the 10 Yardstick Design components:
  - Conceptual Model - Clarify the core concepts
  - Consistency - Plan and maintain both internal and external consistency
  - Content - Fit content to customers who use the product
  - Feedback - Provide reassuring feedback
  - Interaction Model - Clarify the interaction rules
  - Navigation - Structure navigation clearly
  - Terminology - Use plain terminology
  - User Assistance - Optimize user assistance
  - Visual Design - Optimize visual design
  - Contextual - Design for the context of use

The Yardstick for Design

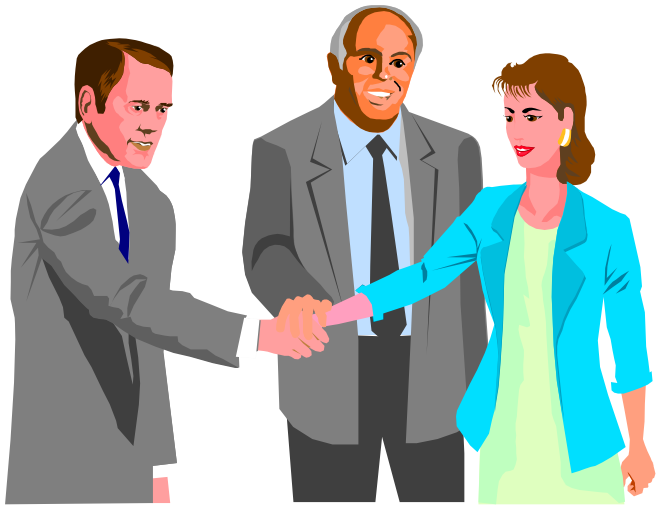
- During this workshop, we will address each Yardstick component except User Assistance.
  - Develop a *conceptual model* based on an object hierarchy that represents the user's tasks and mental model of the domain.
    - The product content and functionality should be organized around the user's objects and tasks.
  - Develop a *consistent* UI framework that represents UI tasks and objects.
  - Develop *content* that embodies the attributes of each UI object and that represents the user's tasks.
  - Begin to develop user *feedback* that supports each task, minimizes user error, and enhances user performance. Detailed user feedback will be developed during Detailed Design.
  - Develop an *interaction* model that allows the user to easily manipulate and manage the set of UI objects.
  - Develop a *navigational* model that allows the user to easily access the functionality and information embodied within the set of UI objects.
  - Develop content that uses *terminology* well understood by users within the product domain.
  - Develop a low-fidelity prototype of conceptual design that begins to mockup layout of UI components and *visual design*.
  - Design for the *context of use*. Develop a conceptual model and low-fidelity prototype supports typical usage scenarios and task flows verified by the users present in the workshop.

Goals of the Roadmap Design Workshop

- Implement user centered, participatory design methods that involve actual users who represent the user profile, key project stakeholders, and technical engineers to jointly produce a conceptual product design.

- Perform iterative analysis, design, and testing to:
  - Identify user needs, requirements, goals, constraints, and tasks
  - Create a set of Task Objects
  - Map tasks and objects into a GUI or Web Browser design
  - Verify usability and accuracy of the user tasks, UI objects, and the Conceptual product design

- Use the outputs of each part of this workshop as input into subsequent parts.
  - User Tasks -> UI Objects
  - User Tasks and UI Objects --> Conceptual Design



Goals of the Three Parts of this Workshop

Part 1: Perform User and Task Analysis

- *Goals:*
  - Identify *User Needs/wants* and *Product Requirements* by performing high-level Task Analysis with key project stakeholders.
  - Build high-level *Task Flow diagrams* and *Task Lists* that represent the User Needs and support the job/tasks that are performed by the user.
  - Perform a “think out of box” group analysis of the task flows to enhance processes and how the job/task is performed.
  - Prioritize functionality from the “user” and “systems” perspective.
  - Make evident the project/design scope (user needs/wants, product requirements, user tasks). Identify functionality and capabilities that are out of scope.
  - Verify that each task flow is complete, accurate and within scope using an informal group walkthrough.

- *Output:*
  - Task Flow diagrams that represent the key tasks performed by the user(s).
    - This diagram is built using index cards to represent process steps and arrow stickies to show data/process flow between the processes.
  - Task Lists with sub-task identified.
  - Use these Task Flow diagrams as input into Part 2 - Task Object definition.

Goals of the Three Parts of this Workshop

Part 2: Map User Needs, User Requirements, and Task Flows to Task Objects

- *Goal:*
  - Map the User Needs/wants, Product Requirements, and Task Flows developed in Part 1 into Task Objects.
    - *Note that these objects are User Objects not System Objects*.
  - Each Task Object represents a discrete unit of information that is manipulated by the user to perform the job/ task.

- *Output:*
  - A set of Task Objects represented as index cards with colored stickies attached that show the Object Attributes, actions, and relationships.
  - Use this set of Task Objects as input into Part 3.

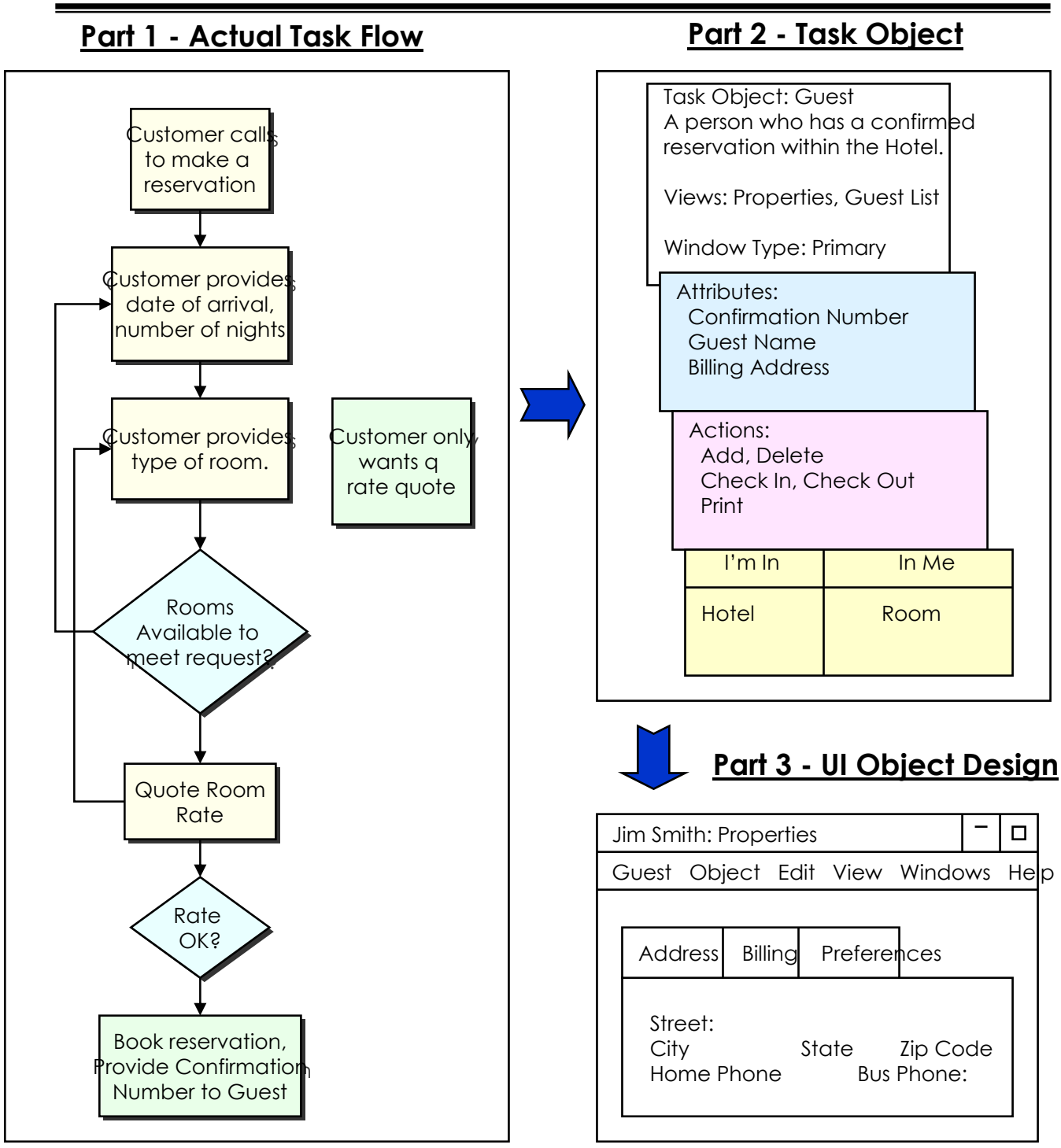
Goals of the Three Parts of this Workshop

Part 3: Map Task Objects to Conceptual and UI Design

- *Goals:*
  - Develop the product Conceptual Design based on the Task Flows, Task Lists, and Task Objects identified in Parts 1 and 2
  - Represent this Conceptual Design into a UI design (e.g., windows, menus, command buttons, and other UI controls).
  - Verify that the UI design is complete, accurate, usable and acceptable to the user(s) by performing informal usability testing.
  - If this is a GUI based product, ensure that the UI design conforms to product/company style guides and the target UI platform style guide (e.g., Microsoft® Windows®).
  - *Note that the default UI framework used by this workshop is based on the IBM™ CUA™ (IBM, 1992) and Microsoft® Windows® look-and-feel.*

- *Outputs:*
  - Low fidelity (paper and pencil) GUI or Browser based design prototype. This design Includes basic window/browser design (e.g., window layout, menus, command buttons, other UI controls).
  - *Note that detailed design must still be performed to complete the UI design (layout fields and controls, author labels/prompts/messages, ensure that all data attributes are represented in the windows, design icons/graphics).*
  - The output from all three parts of this workshop is documented in the *UI Specification* and *Product Style Guide*.
  - *Note that TUG has developed UI Specification and Style Guide templates (MS Word format) to facilitate this documentation*.

Flow of Deliverables Produced in the UCD Workshop



Key Activities for each part of the UCD Workshop

Part 1: *Perform User and Task-Analysis*

- Build the Proposed Work Flow (high-level)
- Perform a Current “as is” work flow analysis
- Identify Problems & Issues in the Current work flow
- Verify the scope of the Current work flow
- Build a Re-engineered version of the Current work flow
- Verify the scope of the Re-engineered task flow
- Build the Actual “to be” Task Flow and Task List using the Re-engineered and Proposed task flows.

Part 2: *Map User Needs, Requirements, and Task Flows to Task Objects*

- Identify Task Objects from the Actual “to be” task flow
- Identity Object Attributes
- Identify Object Actions
- Identify the relationships between Task Objects
- Verify that the Task Objects support the Actual task flow using informal usability testing and task walkthroughs

Part 3: *Map Task Flows and Task Objects to Conceptual and UI Design*

- Determine if the design will be Task or Object based.
- Represent each Object and/or Task as a set of windows.
- Represent Object containment or Task hierarchy
- Represent all Object/task attributes - Build Object views
- Represent the Object/task commands (actions)
- Perform usability testing and task walkthroughs to verify the UI Design

Summary of UCD Workshop Outputs

- The Task Flow diagrams that represent:
  - Proposed Work Flow (high-level)
  - Current Task Flow
  - Re-engineered Task Flow
  - Actual Task Flow

- Task List that represents the Actual “to be” Task Flow

- Task Objects that represent the Actual “to be” Task Flow diagram

- Low-fidelity, paper-and-pencil prototypes of windows that represent each Task Object. This prototype represents the conceptual design of the product.

- UI Specification that documents the design constructed during the workshop.

- UI Style Guide (optional) that documents elements of style/design common across multiple products.

Part 1: Perform User and Task Analysis

(Identify User Needs, User Requirements, and Task Flows)

Part 1 - Overview

- *Goals:*
  - Identify the job/task as is currently performed and the job/task to be performed.
  - Re-engineer the current job/task and processes within the project's scope and timeframe using technology that is either available or that can be acquired/built within the project timeframe.
  - Construct a Task Flow diagram and Task List that best represents the User Needs/wants and product requirements to perform the job/task.

Part 1 - Overview

- Should detailed task analysis be performed?
  - Yes, after this workshop is complete, detailed task analysis must be performed in order to design the backend processes and internal system logic
  - No, detailed task analysis is not needed to develop the UI and Conceptual Design

- Building the Conceptual Design during this workshop, we produce will affect the entire product/application, not just the user interface.
  - We must assess the impact of the Conceptual Design and UI on the Database and System design. Verify with the technical engineers that this conceptual design will work.
  - We must design to accommodate known technical and business constraints
  - We must design to accommodate existing product and system components that will not be replaced.

Part 1 - Overview

- *Inputs:*
  - The collective knowledge brought into the workshop by each user and stakeholder.
    - This knowledge may include product vision, user and business needs, business rules, and technical constraints (e.g., system, UI, database).
  - User artifacts and product/job documentation may be brought to the workshop as examples of current work and to ensure that job/task constraints are well understood by the team.
    - forms, reports
    - business rules, job manuals, standards

- *Outputs:*
  - Task Flow diagram(s) and a Task List that best represents the User Needs/wants and Product Requirements to perform the job/task.
    - The task flow should contain sufficient detail to describe each step performed by the user.
    - Each task step should contain one verb and one noun.
    - Make evident in each task step which user profile(s) perform the task. Also note when an output is used as input by another user profile within the scope of this project.
    - These task flows are represented as a flow chart of index cards and sticky arrows.

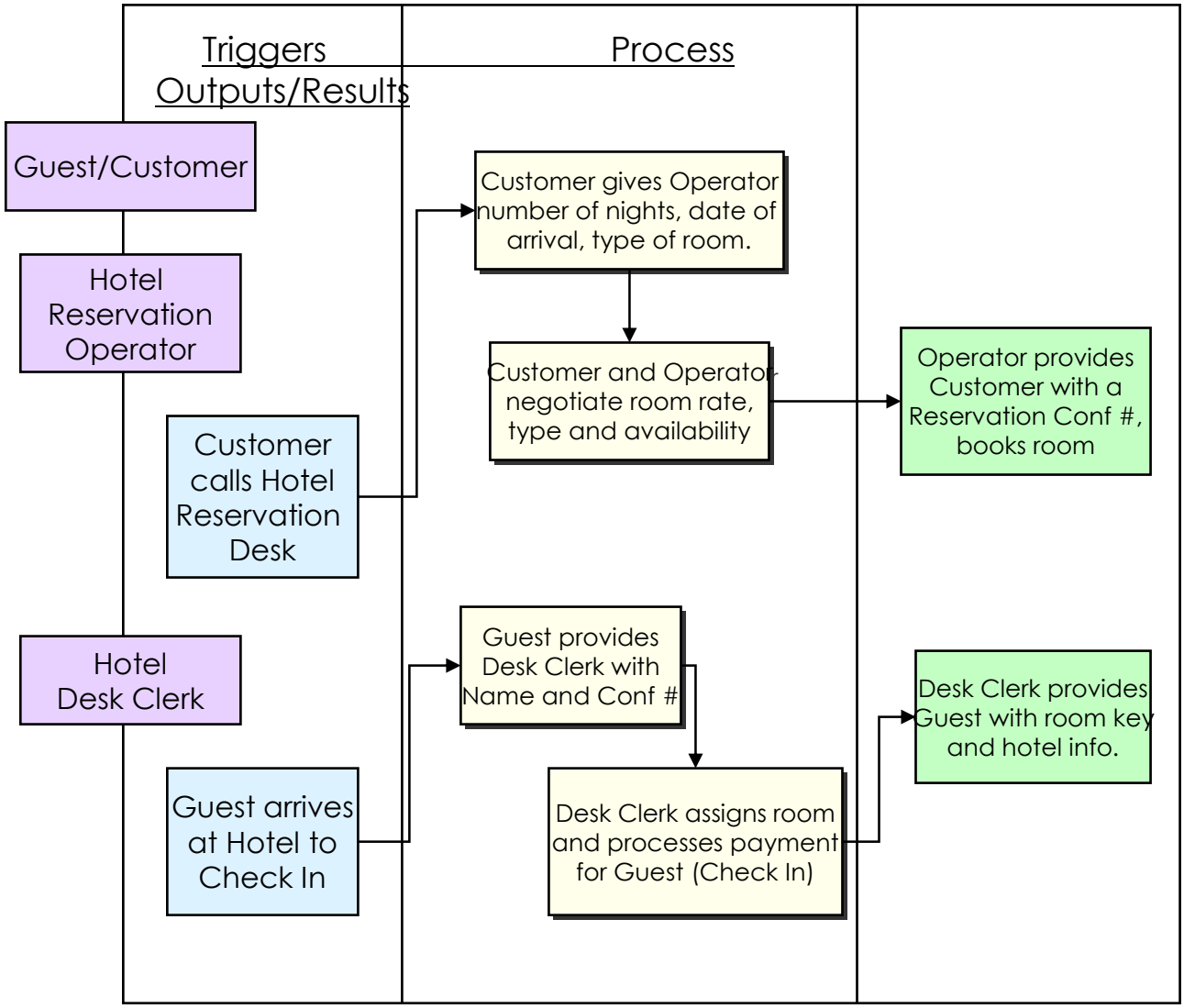
Proposed (Envisioned) Task/Work Flow

- *Goal:*
  - At a high-level, define the job or task to be performed by the user(s) with this product. This is the "proposed" and "envisioned" task flow.
  - Ensure that the product vision is represented.
  - Capture the envisioned, not the as is. The "as is" is captured in the next step.

- *Output:*
  - High-Level Proposed Work Flow should capture:
    - Context in which user's job is performed
    - Inputs to and outputs from the user's job/task
    - Major process steps (the task steps) to produce the output.

- *Exercise:* Build a high-level flow diagram that best represents the "to be" work flow of the user(s). Use yellow index cards in this exercise. Attach each index card to a large piece of flip-chart paper.
  - Record each major Task/Process step on a separate index card..
  - Record each user involved in performing these tasks on a separate index card - Attach the user cards along the left-hand side of sheet.
  - Record each Trigger (an event which initiates a user to perform a task - phone call, e-mail) on a separate index card.
  - Record each Result/Output (whatever is produced as a result of performing the task) on a separate index card.
  - Use sticky arrows to connect the index cards as needed to designate flow between each of the task steps.

Proposed Task/Work Flow - Example

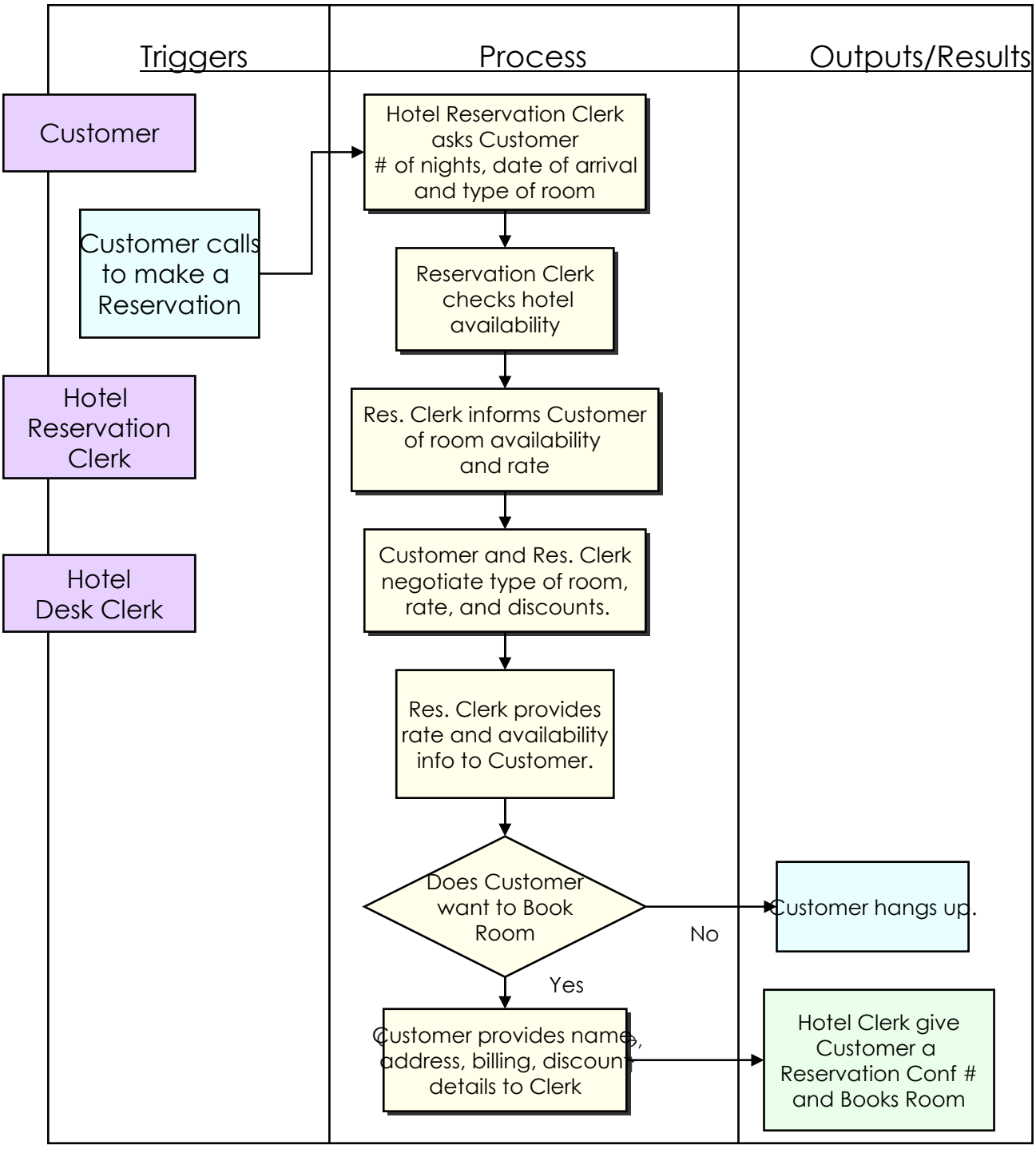


Current (As Is) Work Flow

- *Input:*
  - Proposed Work flow that represents the high-level view of the job/task to be performed.
- *Output*:
  - Build a new Task Flow using a moderate level of detail to define the process/task steps currently performed by the user to perform the job/task captured in the High-Level Proposed Work Flow (may be manual or automated). This is the as is work flow.
  - Only a moderate amount of detail is required since we will continue to iteratively refine and define the task detail.
  - Record issues about any step in the work flow on a neon sticky and attach it to the appropriate index card.

- *Exercise:* Document in a moderate amount of detail the way the job/task is currently performed by the user(s). Use yellow index cards in this exercise. Attach each index card to a large piece of flip-chart paper.
  - Record each task/process step performed by a user on a separate index card.
  - Record the User Profile name of each user performing a task on a separate index card. Attach these cards along the left-hand side of the flip-chart paper.
  - Use sticky arrows to represent the data/process flow between the index cards.

Current (As Is) Work Flow - Example



Problems & Issues in the Current Work Flow

- *Input:*
  - Use the Current Task Flow card layout (yellow) produced in the previous step.

- *Output:*
  - Current Task Flow card layout (yellow) with *Problems & Issues* stickies attached to the appropriate card.

- *Exercise* - Identify and highlight all Problems & Issues in the Current Task Flow.
  - Briefly discuss as a group all issues and problems with the current task flow.
  - Record each issue and problem on a separate raspberry neon color stickies. Attach each sticky to the appropriate yellow index card.
  - Have the user(s) rank each issue by importance to them (e.g., High, Medium, Low). Record the level of importance on each sticky. Take the **users'** point of view.
  - By ranking the issues, the group will have a better understanding of which issues to address during the design re-engineering step.

Review the Scope of the Current (As Is) Work Flow

- *Input:*
  - Use the Current Task Flow card layout (yellow) with Problems & Issues identified and prioritized on raspberry neon stickies.

- *Output:*
  - Review the scope of the Current Workflow for this workshop. Identify the highest priority components of the Current Task Flow.

- *Exercise*:
  - Review the scope (tasks, functionality, outputs, technical issues) of the Current Work flow for this workshop. Consider the following:
    - Problems & Issues (Importance? Level of complexity? Must these problems be resolved to implement the design?
    - Are the subject matter experts and engineers needed to complete this design participating in this workshop? Are they available to be brought-in as needed?
    - Is this group authorized to design the components of this workflow?
    - How many days have been allocated to complete this workshop?
  - Use a black marker to circle and label the portion of the Current Work Flow that will be designed in this workshop. Out of scope components may be designed in subsequent workshops.
  - Hang-up the Current Work Flow chart (yellow cards) for reference throughout the workshop.

Re-engineer the Current Task Flow

- *Input:*
  - The “in-scope” portion of the Current Task flow layout (yellow) with the Problems & Issues stickies attached to the appropriate index cards.

- *Output:*
  - Re-engineered Task flow, defined at moderate level of detail. Use a blue index card to record each task step.
  - **The Re-engineered task flow should resolve the issues indicated on the neon raspberry stickies.**

Re-engineer the Current Task Flow

- *Exercise*: Build the Re-engineered Task Flow:
  - Re-engineer the Current Work Flow to solve each Problem and Issue. Construct the Re-engineered task flow using a moderate level of detail.
    - Brainstorm solutions to each Problem and Issue. Follow the standard set of rules for a Brainstorming session.
    - Record concerns on a raspberry neon sticky and attach to the back of the appropriate blue card.
    - Use the priority levels set for the Current Work Flow Problems & Issues to help drive the Re-engineered design.
    - In cases where the Re-engineered task flow is currently unachievable, it should be achievable within the next 1-2 years.
  - Build the Re-engineered Task Flow to represent the ideal way the job could be performed. Use a blue index card for each task step. Attach each index card to a large piece of flip-chart paper.
    - Copy each trigger from the Current Work Flow onto cards.
    - Record each step/task on a separate index card using a moderate amount of detail.
    - Use sticky arrows to show the process flow and data between cards
    - Record each output/result on a separate card.

Prioritize the Re-engineered Task Flow Steps

- *Input*:
  - Re-engineered Task flow from the previous step

- *Output:*
  - Re-engineered Task with each task step prioritized by “User Priority” and “Ease of Implementation”.

- *Exercise* - Prioritize each step of the Re-engineered Task Flow.
  - The user(s) should prioritize the importance and need for each step in the task flow.
    - Use High, Medium, and Low ratings.
    - Use an orange neon sticky to each record the *user priority* for each step in the task flow.
  - The technical engineer(s) should estimate the ease of implementing each step in the task flow.
    - Use Easy, Moderate, and Hard ratings.
    - Use a blue neon sticky record the *implementation difficulty* for each step in the task flow.
  - Build a legend for both the User and Engineer ratings. Record the legend on a card and attach to the task flow layout.
  - As a group, decide
    - which Re-engineered steps should be carried forward into the conceptual design
    - which steps are already implemented in the Current Task Flow
    - which steps cannot be carried forward into the product (not feasible, no time to implement).

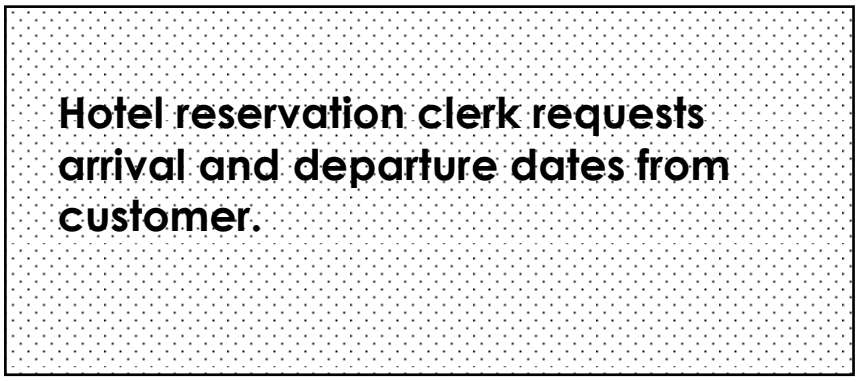
Actual (To Be) Task Flow

- *Inputs:*
  - Proposed Work Flow and the prioritized (user and engineer priorities)
  - Re-engineered task flow

- *Output:*
  - Actual Task Flow layout that will serve as the basis for the Conceptual design in Parts 2 and 3 of this workshop.
    - The Actual task flow should take into consideration the “to be” vision for the product, the Re-engineered “as is” task flow, and the priorities defined by the group.
    - Ensure that all functionality included in the product vision, but not part of the user’s existing task flow is included.
    - The Actual task flow should include a sufficient level of detail to carry forward into Part 2 of the workshop.
  - Task Lists based on the Actual Task Flow

Actual (To Be) Task Flow

- *Exercise*: Build the Actual and Detailed Task Flow. Use white index cards for any new steps built in this exercise. Attach all cards to a large piece of flip-chart paper.
  - Redesign the Re-engineered task flow based on the user/engineer ratings from the previous step. Each step of this new task flow should be achievable using resources available to the project.
  - Include sufficient detail within the task flow to accurately describe the steps that must be performed by the user(s). Each task card should have the following:
    - description of the task
    - a noun and a verb as shown below
    - who is performing the task
  - If any of the steps remain the same as in the Current or Re-engineered Task Flow, carry forward the yellow and blue cards instead of creating a new white card. This will help highlight the difference between the "as is" and "to be" task flows.
  - Use sticky arrows to show the flow of data/process between the cards.



Review the Scope of the Actual (To Be) Task Flow

- *Input:*
  - Actual “to be” Task Flow from the previous step.

- *Output:*
  - Review the scope of the Actual Task Flow for the remainder of the workshop.
  - Identify which components are “in scope” and which components are “out of scope”.

Review the Scope of the Actual (To Be) Task Flow

- *Exercise*:
  - Review the scope of the Actual "to be" Task Flow (project components, technical issues) for the remainder of the workshop. Consider the following:
    - Is the Actual Task Flow within the project scope?
    - Are the resources available to complete design and construct the task flow within the expected project timeframe?
    - Are the infrastructure, database, and back-end components available or can they be built/acquired within the project timeframe?
    - Problems & Issues (Importance? Level of complexity? Must these problems be resolved to implement the design?
    - Are the subject matter experts and engineers needed to complete this design participating in this workshop? Are they available to be brought-in as needed?
    - Is this group authorized to design the components of this workflow?
    - How much time has been allocated to complete this workshop?
  - Use a black marker to circle and label the portion of the Current Work Flow that will be addressed in the balance of this workshop. Components outside this scope can be handled in subsequent workshops.
  - Hang-up the Actual Work Flow chart for reference throughout the workshop.

Perform a Usability Walkthrough of the "To Be" Task Flow

- *Goal*:
  - Ensure that the usability of the Actual "to be" Task Flow is acceptable to the user(s) and that the Actual Task Flow is complete and accurate.
  - Determine if steps are missing or if steps included are not needed.

- *Input*:
  - Actual "to be" Task Flow layout

- *Output*:
  - Actual Task Flow layout that has been tested and modified as necessary.

- *Exercise*: Perform an informal usability walkthrough of the Actual Task Flow to test the accuracy, completeness, and acceptability of the layout.
  - Have each user perform the task walkthrough. The user should point to each card and describe the step (What actions are required? What data is needed? What decisions are made?).
  - All team members should observe and help identify missing steps, incorrect logic, any inaccuracy, and ambiguity in the task flow.
  - Modify the task flow as needed to correct any problems. Add cards as needed. Change flow arrows as needed.
  - Retest the design

Build a Task List

- Goal: Build a Task List
- Input: Use the Actual “To Be” Task Flow
- Output: Task List that includes the steps “to perform” each task.
  - This Task List will be used in Part 3 of this workshop to build the Conceptual Design.

- Exercise: Build a Task list
  - Look for the activities (verbs) that users want to accomplish in the Actual Task Flow. Record each activity on a separate Green index card.
  - Identify the steps (the how) to perform each task. Record these steps on a separate White index card.
  - List other ways to perform the task under Alternate Steps. Record alternative ways to perform the task on a White index card.
    - This is an optional step.
    - Variations may be required to build a more flexible tool.
    - Variations tend to complicate UI design.
    - All variations must support the task and user goals.
  - Attach all index cards to a Task List Table using a large sheet of flip-chart paper.

Build a Task List

Task	Task Steps	Alternate Steps

Part 2: Map User Needs, User Requirements, and Task Flows to Task Objects

Part 2 - Overview

- *Goal*:
  - Map the User Needs/wants, Product Requirements, and Task Flows from Part 1 into abstract Task Objects.

- *Input:*
  - Actual “to be” Task Flow from Part 1.

- *Output:*
  - A set of Task Objects that support execution of the Actual “to be” Task Flow defined in Part 1.
    - The set of Task Objects will be translated into a Conceptual Design and UI in Part 3.
    - We will not begin to construct the UI until Part 3.
    - The set of Task Objects are platform independent and may be represented using either a browser or traditional GUI style.

What is a Task Object?

- A Task Objects are:
  - Abstract representation of data and actions.
  - Discrete unit of information with behavior and containment relationships with other objects.
    - The data, actions, behavior, and containment of the Task Objects should be natural to the job/task and represent the user's mental model of the job/task.
  - Managed and manipulated by the user to execute the task flows.
  - "User objects" and should not be confused with System Objects.
    - User objects represent data, process, and logic within the product.

- The set of Task Objects and their relationships (object hierarchy) should match the user's mental model of the job/task.

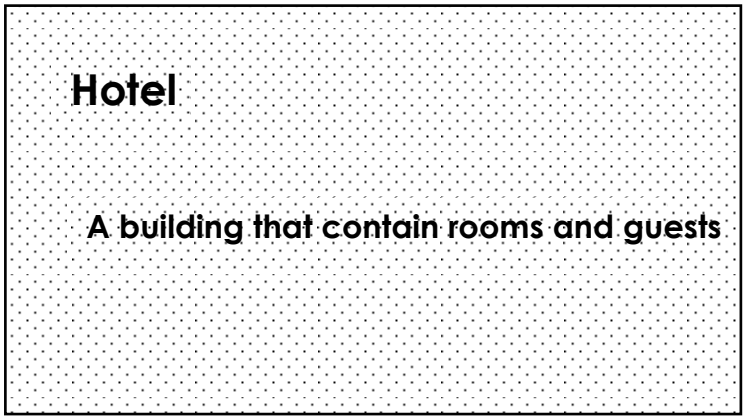
- At this point, don't worry about the number of Task Objects. As we proceed through Part 2, we will eliminate all task objects that are simply attributes of other objects.
  - Keep in mind that the complexity of your UI is proportional to the number of task objects.

Identity the Task Objects

- Goal:
  - Identify the Task Objects within the Actual “to be” Task Flow.

- Input:
  - Actual Task Flow from Part 1.

- Exercise: Extract all potential Task Objects from the Actual Task Flow.
  - As a group, find the nouns in the Actual Task Flow. Note: Verbs will be recorded later.
  - Record each noun on a separate white index cards. Each of these nouns is a potential Task Object.
  - Review all the index cards and discard the duplicates.
  - Add a brief description to each remaining object card. This description should be accurate and within the context of the task flow.



What is a Task Object Attribute?

- Each Task Object must contain 1 or more Attributes that represent data.

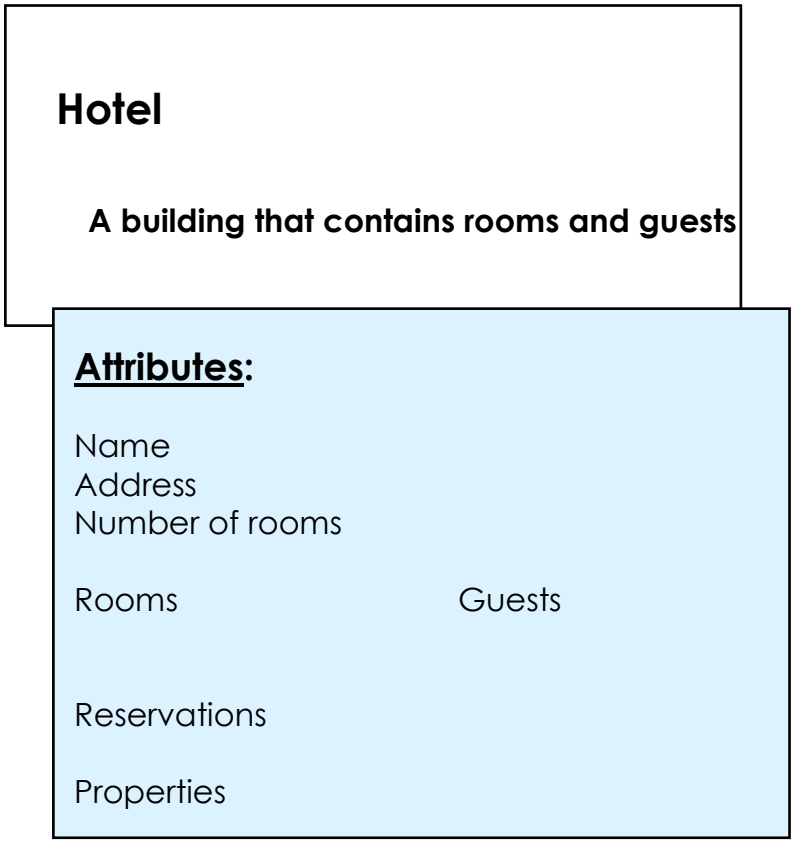
- Objects may have the following types of attributes (e.g., What the object *is made up of?)*:

  - Discrete units of data that describe the object. For example:
    - *Name:* Buckhead Hotel
    - *Location:* 1500 Peachtree Road, Atlanta, GA
    - *Number of Rooms*: 150
    - *Status:* Vacancy

  - Child object(s) - attributes that represent data contained by another object. For example:
    - Rooms
    - Guests
    - Reservations

Define the Attributes for each Task Object

- Exercise: For each Task Object card, define an initial set of Attributes.
  - Record the set of Attributes for each Task Object on a light blue sticky. Attach the blue sticky to the task object index card.
  - Set aside all Task Object cards that do not have a blue sticky. Do not discard these cards. *Note: If a Task Object has no attributes, it is probably not an Object and will likely become an attribute of another object.*
  - Review the remaining Task Object cards you set aside. Determine if any of these card(s) should be attributes of another Task Object.



Task Object Actions

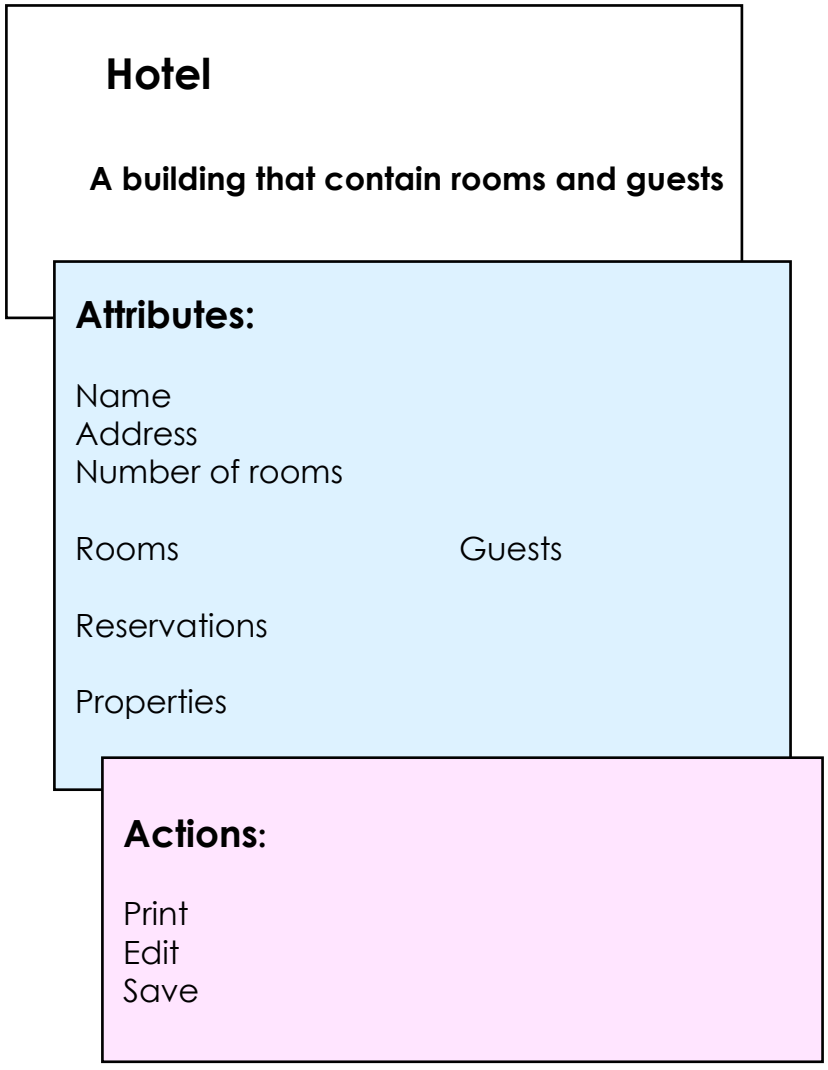
- Each Task Object has a set of associated actions (e.g., Create, Modify, Delete, Print, Run).

- In the previous step, we extracted the nouns from the Actual Task Flow to produce the initial set of Task Objects. In this step, we will go back pull out the verbs in the task flow. For example, “Reserve”.

  - In the task step “Reserve Room for Guest”, *Room* is the Task Object and *Reserve* is an Action.

Define the Actions for each Task Object

- *Exercise*: Review the Actual Task Flow to identify all the verbs that may be Actions associated with a Task Object.
  - Record the actions for each Task Object on a pink sticky.
    - For example, can the object be.. Created? Modified? Deleted? Saved? Printed? Run?
  - Attach the Pink sticky to the Attribute sticky of the associated Task Object.



What are the Types of Task Object Attributes?

- An attribute may be a discrete unit of data or another Object.
  - *Properties* - An attribute is a property when it is a discrete unit of data that is not treated as an object within the context of its parent.
  - *Child Objects* - An attribute is a Child Object when it is treated as an object within the context of its parent.

- Below are examples of attributes for the Room and Guest Objects in the Hotel example:

Task Object: Room

Description: A physical location within a Hotel that may be rented to a Guest.

Attributes:

Number
Status (reserved, available, not available)
Price

Guest

Properties

Task Object: Guest

Description: A person who has a confirmed reservation within the Hotel.

Attributes:

Name
Billing Address
Confirmation Number

Room

Properties

What are Child Objects?

- An object is a Child Object when it is an attribute of another object.
- The “parent” and “child” relationship within and between task objects refers to the logical hierarchy of the object set. This logical hierarchy is how the user perceives the relationships between the objects and does not reflect an underlying system or data model.
- The object hierarchy should represent the user’s mental model of the “user” objects and support the user’s need to access data and functionality within the Task Flow.
  - When you design the object containment hierarchy, you are designing the foundation of the users’ mental model for their domain.
  - The objects and object hierarchy should be familiar to the user and represent how the user thinks about the job/task.
  - For example,

    House

    Rooms

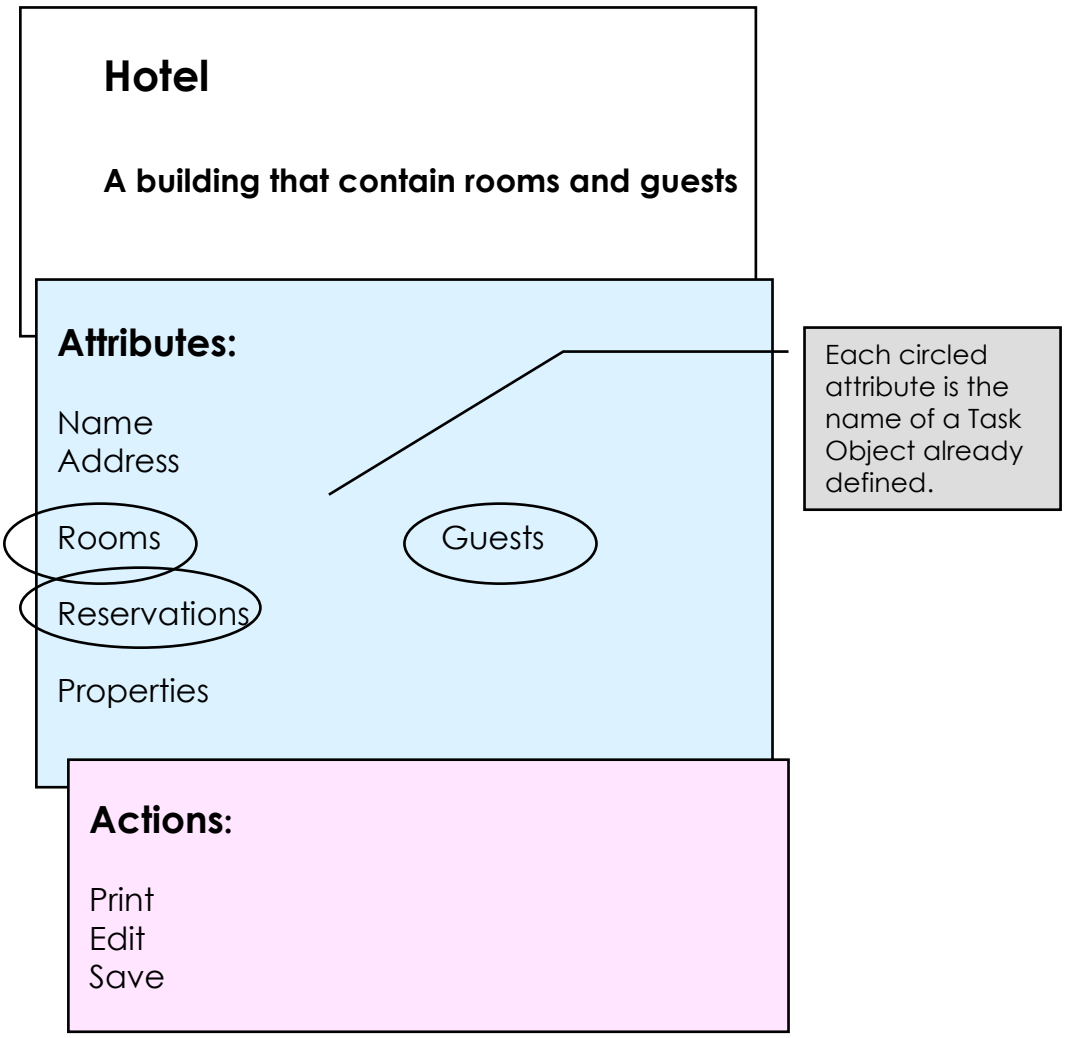
    Closets

    Clothes

- An object can be a “child object” of one object, and a “property” of another object.

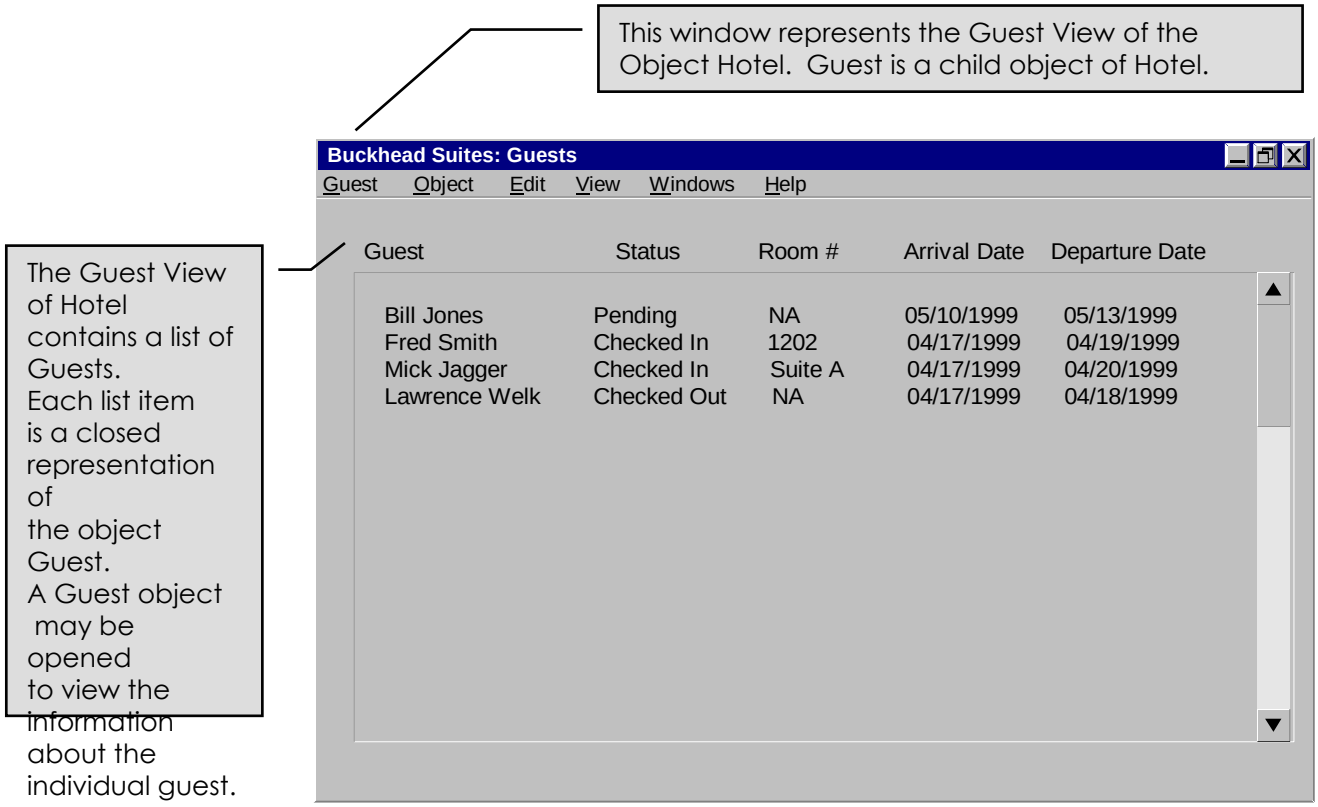
Circle the Attributes that are Child Objects

- *Exercise*: For each Task Object, identify the Attributes that are themselves defined as objects (have their own white index cards).
  - Circle these objects on the blue Attributes sticky.
  - The circled attributes will likely become *child objects* of this task object. The other non-circled attributes will become *properties*.



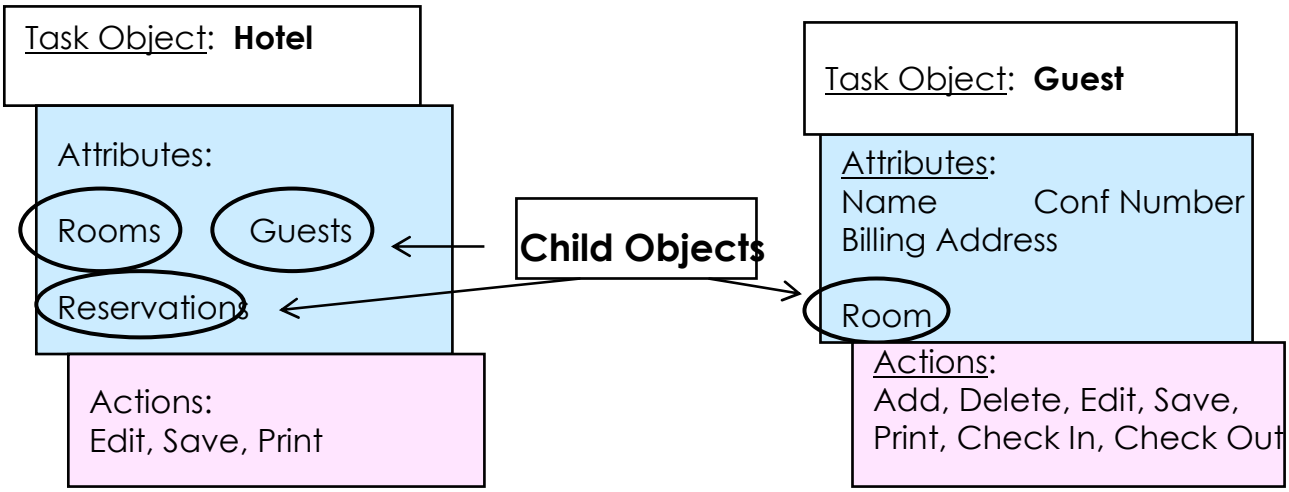
What is Task Object Containment?

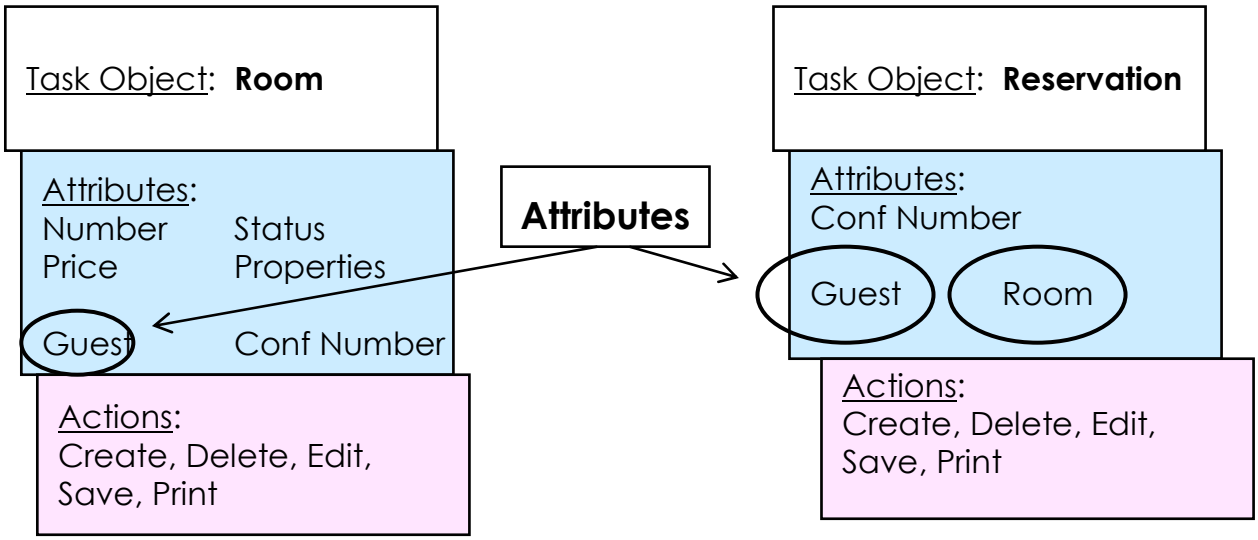
- *Goal:*
  - Define the containment relationships between the Task Objects. These relationships represent the hierarchy of Task Objects and become the navigational model of your UI. This hierarchy should represent the user's mental model of the tasks performed and artifacts manipulated, as well as the real-world relationships between the objects in the user's domain.
- *Input:*
  - Task object cards and stickies with the attributes circled from the previous step,
- *Output:*
  - Task Object Containment relationships and hierarchy.
  - From the Hotel example, the following UI design shows the parent and child relationship between Hotel and Guest. Guest is a child object of Hotel. This design shows the Guest view of Hotel where each list item is a Guest object in its closed state.



Task Object Containment

↗ Below are examples of object containment for the Hotel example. The optimal and correct hierarchy of these objects must still be defined.

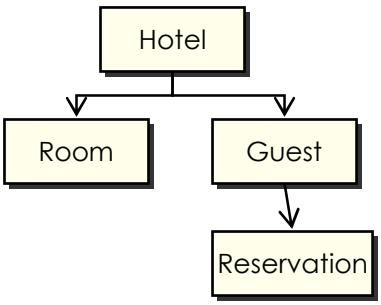




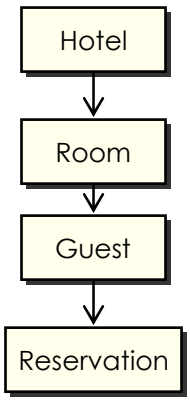
Optimize the Object Hierarchy

- The hierarchy for the entire set of Task Objects is considered optimized when all tasks in the Actual Task Flow are efficiently supported.
- It is critical to ensure that the UI Object model is optimized to address functionality to be added in future releases of the product.
  - Each product or application should have a single UI object model.
  - UI Object model may be extended to support new objects not considered in this workshop.

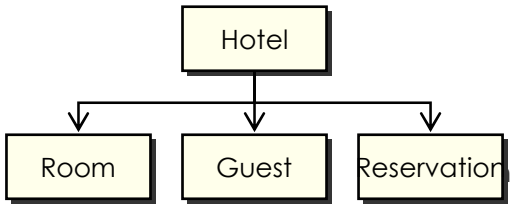
- Hierarchy #1 is optimized to view available Rooms and Guests who have Checked In or have a reservation. However, you cannot access a list of reservations without first going through a Guest.



- Hierarchy #2 is optimized to view a list of available rooms or room facilities. You cannot access a Guest until that Guest has checked in and has a room # assigned. Likewise, reservations cannot be obtained until the Guest checks into a Room. A manager could not obtain a list of reservations for the day.

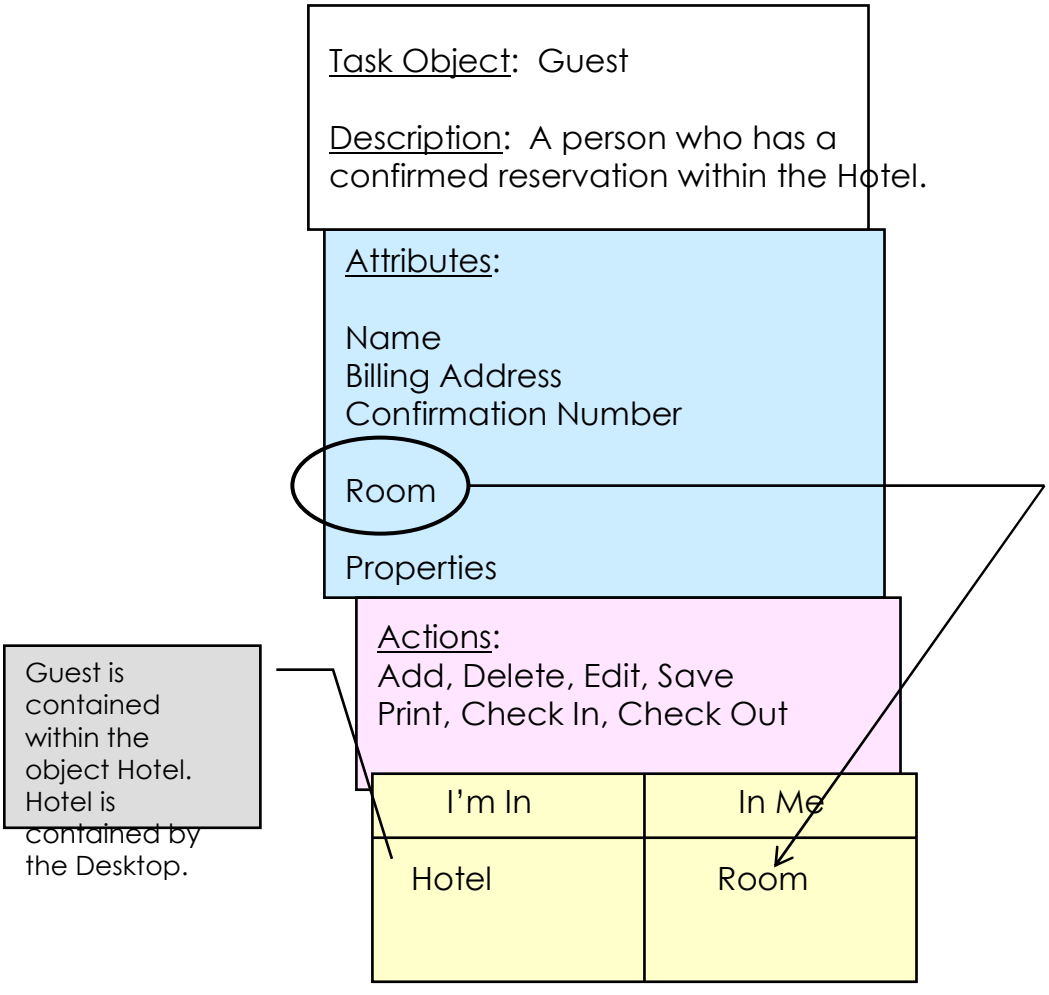


- Hierarchy #3 is globally optimized for tasks involving Room, Guest, and Reservation.



Task Object Containment

- Exercise: Resolve the containment relationships for each Task Object.
  - For each object, add a yellow sticky with 2 columns: I'm In and In Me
    - I'm In column - Record the names of each object that has this object's name circled as an Attribute.
    - In Me column - Record the names of each object circled in this object's Attribute list.



Task Object Containment

- *Exercise Cont'd*: As a group, iteratively refine the object hierarchy until everyone is satisfied that the logic is correct.
  - Throughout this exercise, keep in mind that the containment model is based on the user's "real-world" mental model of the objects and the relationships between task objects.
  - Review all the circled Attributes to ensure that they are actually child objects and not properties.
  - Review all the uncircled Attributes to ensure that they should not be Child Objects rather than properties. Circle all properties that should be Child Objects and update the I'm In and I'm Me columns for each Task Object.
  - Uncircle any Child Object you later decide is only a property.
  - Create new Container Objects as needed to better organize the set of Task Objects.
    - Be sure to complete the Attributes, Action, and Containment stickies for each new Container Object.
    - Go back and update the containment stickies for each task object card if your new objects change the object hierarchy.

- Use the following rules to ensure you object hierarchy is correct:
  - Each object can only have one immediate parent
  - A child object cannot be the parent of its own parent.
  - An object may not contain its own parent as a child object (i.e., circled attributes)
  - An object may contain the parent object's name as a property.

Usability Testing Task Objects Against the Task Flow

- Ensure that the set of task objects allows the user(s) to easily and efficiently perform the Actual Task flows defined in Part 1.
  - Usability must be acceptable to the user(s).
- Fix any Task Objects or Task Flows found to be incorrect or incomplete.
  - Since this is an iterative design process, we will continue to make changes and verify our design throughout this workshop.
- Keep in mind that you are testing the underlying conceptual basis of your design, not the User Interface. Ensure that the set of Task Objects and the object hierarchy support all the tasks.

- *Exercise*: Perform an informal usability test of the Object Hierarchy.
  - The user(s) should walk through each of the task flows using the Task Objects to perform the tasks.
  - Each team member should observe carefully to verify that the objects, attributes, and actions recorded on the index cards and stickies are complete and correct. Also, ensure that the Task Flows are complete and correct.
  - If you discover that a part of the task cannot be performed because of a missing task object, create a new task object.
  - Likewise, if you discover that a task object is not an object, but only an attribute of another object, change all the task object cards accordingly.
  - If the task objects you have created help you rethink the Actual Task Flow, change the task flow accordingly.

Part 3 — Mapping Task Objects and Task Lists into a Conceptual Design

Part 3 - Overview

- *Goal:*
  - Translate the Task Objects and Task List from Parts 1 and 2 into a Conceptual Design prototype.
  - Design prototype windows (menus, command buttons, fields, dialogs, and other UI controls) that represent the Conceptual Design and support the Actual Task flow built in Part 1.

- *Input:*
  - Task Objects (attributes, actions, and containment) defined in Part 2.
  - Task List defined in Part 1.

- *Output:*
  - Low-fidelity, paper and pencil UI prototype that includes:
    - Window Type (primary or secondary)
    - Window Views (high-level window layout for each view)
    - Pull-down menu structure
    - Command buttons
    - Key UI controls used throughout the design (e.g., tab controls, Explorer style navigation)
  - The conceptual UI design prototype is tested for usability and completeness using the Task Flows from Part 1.

- *Notes*:
  - Areas of GUI design not directly addressed in Part 3 include:
    - Mouse & keyboard interaction details (e.g., accelerators, mnemonics, drag-and-drop, selection policies)
    - Detailed window design (e.g., prompts, labels, placement of controls)
    - Graphics and Icon design
    - User Assistance (e.g., Tutorials, on-line help, documentation)
  - The default GUI framework used in Part 3 of this Workshop is based on the IBM CUA (1992) and Microsoft Windows® (1995) style guides.

Task vs. Object based Design

- Object based UI Model
  - UI organized around User Objects
    - Tasks (refer to the Task List) are organized within the UI Objects
  - Each UI Object is represented by a set of windows and dialogs
  - Actions that can be applied on each object are represented as buttons and menu selections
  - Tasks are performed by viewing object content, manipulating object content, and applying object actions.

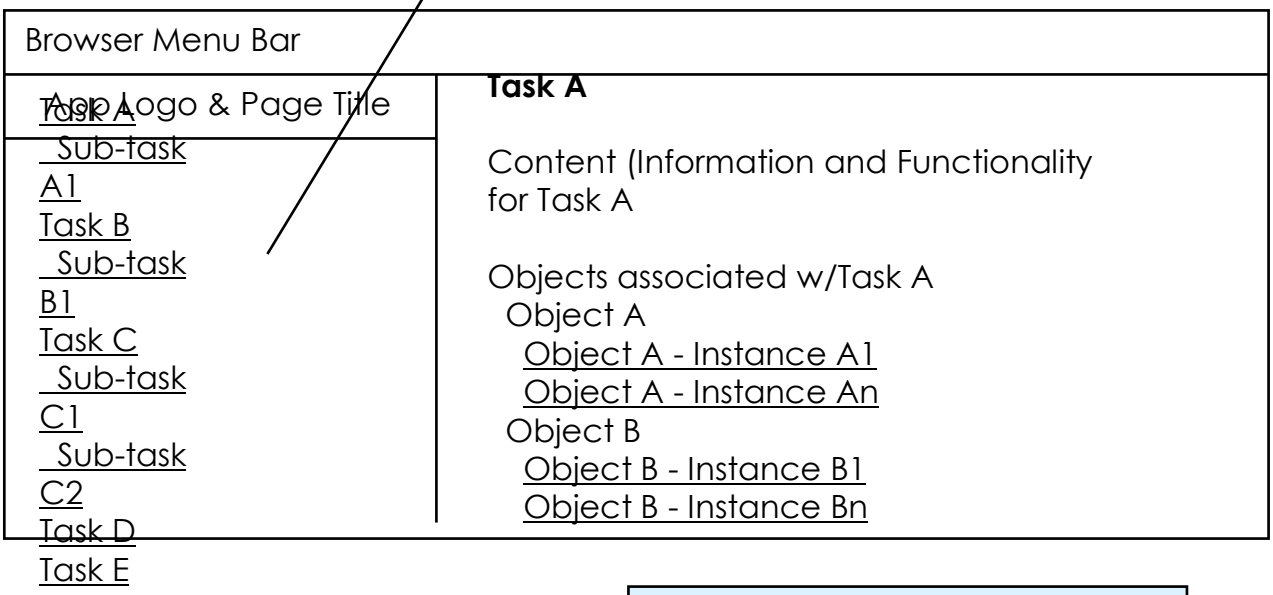
- Task based UI Model (procedural design)
  - UI organized around User Tasks (refer to the Task List)
    - UI Objects are organized within the Tasks
  - Each task is represented by a set of windows and dialogs
  - Task related commands and actions are represented by command buttons and menu selections.

Task Model vs. Object Model UI

elow are examples of Object and Task based UI design in a Browser style interfa

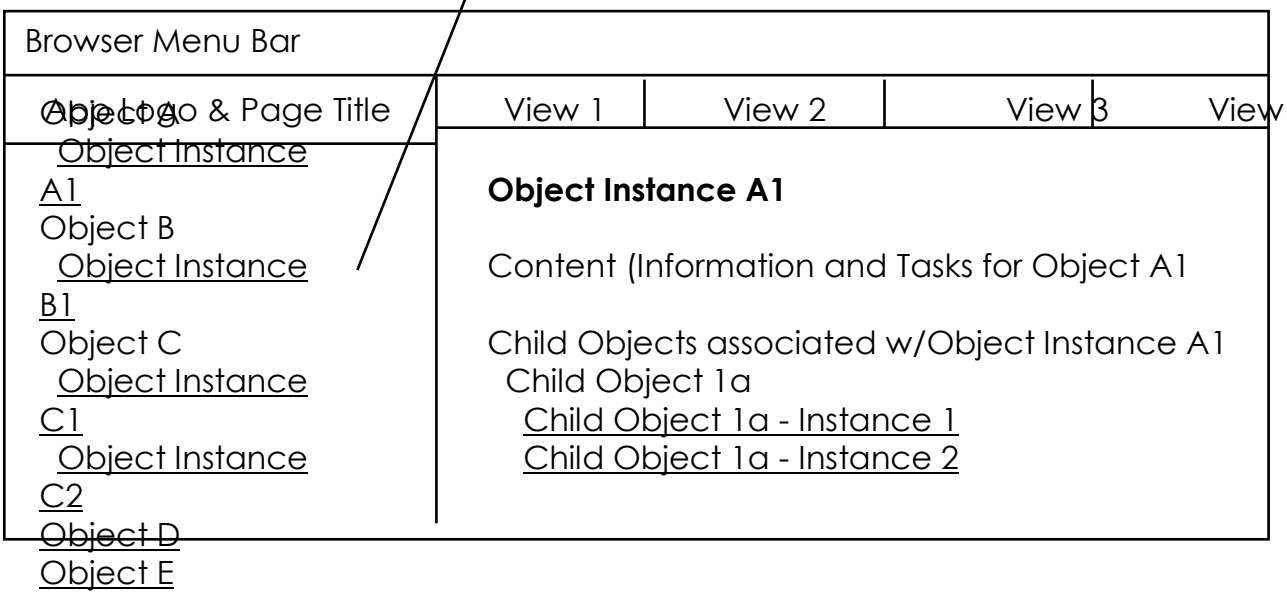
Task based Design Model

Tasks from Task List Table



Object based Design Model

UI Objects from Part 2 of the Workshop



An Introduction to Object Oriented UI Design

- What is an Object Oriented User Interface?
  - An OO UI is a representation of a set of objects that depict real-world units of information commonly manipulated by a typical user in performing a task/job.
  - The set of UI Objects and the Object hierarchy should match the user's mental model of the task/job and facilitate learning of the product.
- The basis for an OO UI design includes:
  - A set of Objects
  - Relationships between these Objects (hierarchy)
- The navigational model is based on the Object hierarchy.
- The resulting OO UI design:
  - Reinforces the Object hierarchy and relationships
  - Allows the user to view and manipulate the UI object hierarchy.
  - Allows the user to access multiple views of an UI Object.
- An OO UI uses a consistent set of rules for defining and managing (e.g., creating, manipulating, and destroying) UI objects and windows within and across software products. This set of rules is typically referred to as the *UI Framework*. This *UI Framework* may be Browser or GUI based.

An Introduction to Object Oriented UI Design

- What are the differences between an Object Oriented UI and Object Oriented Design and Programming?

  - *Objects*
    - An object refers to a User (UI) Object. It is a unit of information that the users commonly manipulates.
    - UI objects do not necessarily correspond to objects within the System Object Model.
    - UI objects are abstract in nature and are not directly represented in code.

  - *Object Hierarchy*
    - UI object hierarchy is defined by the containment relationships between the set of UI objects.
    - UI object hierarchy is a visual representation of the conceptual and interface design displayed to the user and does not represent an underlying Object Model.

  - *Object Inheritance*
    - The user may create a new instance of a UI object class (e.g., Guest), but that instance does not inherit any attributes from the generic Class (e.g., Guest).
    - The user may perceive inheritance from their understanding of the UI Object class (e.g., Guest).

Why implement an Object Oriented UI?

- Why use an Object Oriented User Interface?
  - Flexibility - an OO UI provides the user with a flexible tool to manage and manipulate objects and data.
  - Consistency - an OO UI uses a single UI Framework as a basis for design. Learning is incremental.
  - Expandability - a well designed object hierarchy may be easily expanded to include new functionality. Some functionality may be provided by simply adding another object view.

- Why NOT use an Object Oriented User Interface?
  - No procedural guidance for the task, relies heavily on task based help.
  - User must learn the UI Framework.

Task Model vs. Object Model UI

- Use the following decision table to help determine if the design should be Task or Object based.
  - For each Yardstick Element, rate using both an Object and Task based model to implement your design. Use a 1-5 rating scale.

Yardstick Element	Base UI on an Object Model	Base UI on a Task Model
Conceptual		
Content		
Consistency		
Feedback		
Interaction		
Navigation		
Terminology		
User Assistance		
Visual Design		
Context of Use		
Total score		

Definitions of GUI Objects and Elements

↗ *Task Objects*

- Task Objects are representations of data and actions that are familiar to the user and may be manipulated to execute the task flows.
- In our Hotel example, Rooms, Guests, and Reservations are examples of Task Objects.
- These Task Objects are defined within the system as data stored in one or more databases.

↗ *UI Objects*

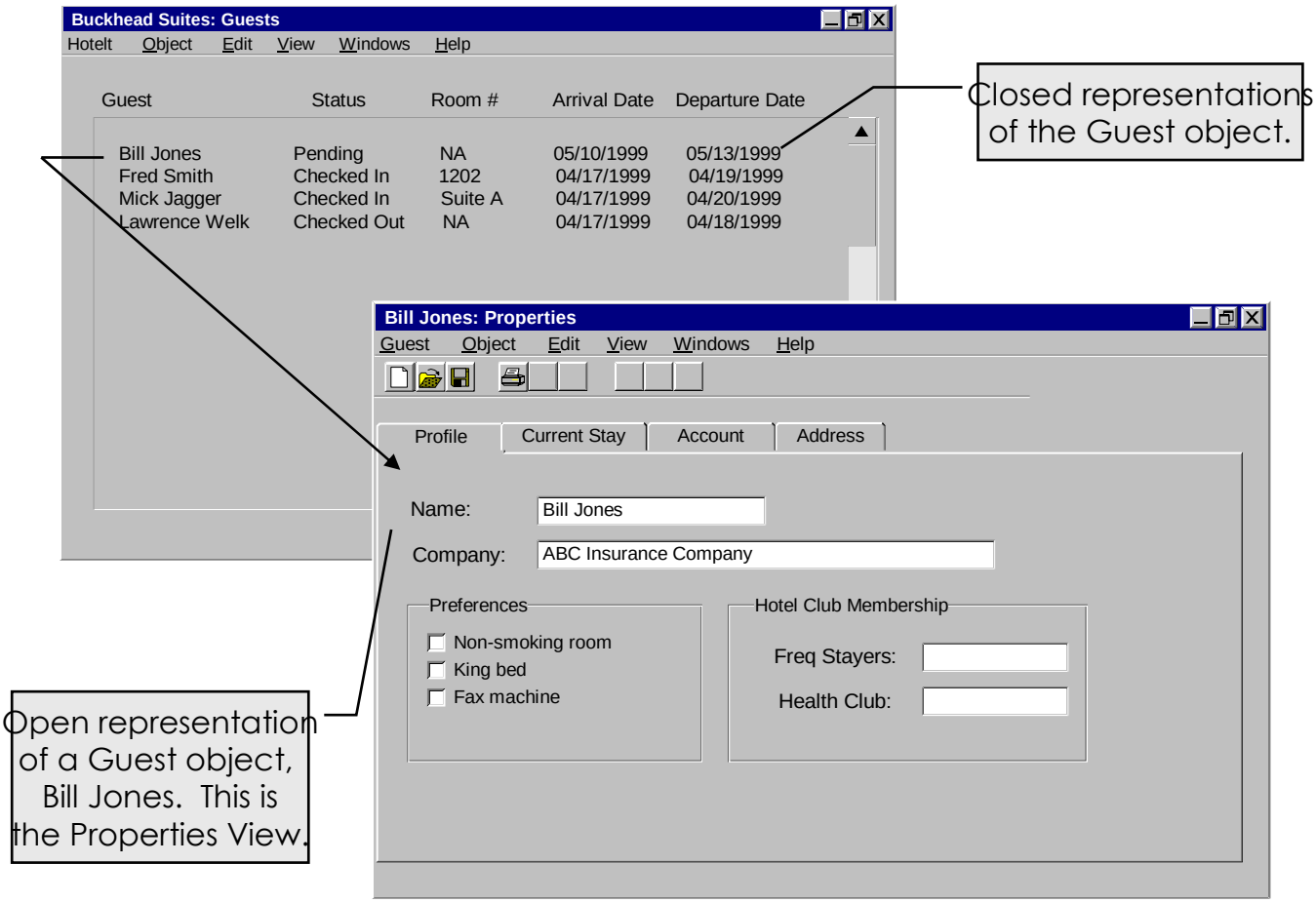
- UI Objects are representations of data and actions that are displayed in the UI.
- UI Objects are represented by UI elements (e.g., windows, icons, list items)
- UI Objects may be independently managed (e.g., created, modified, saved, printed) and manipulated (e.g., selected, copied, moved).
- Objects in a GUI Desktop may include Directories and Files. Directories may contain sub-directories. Both Directories and Files have properties.

↗ *UI Elements*

- UI Elements are any control provided by a User Interface Platform (e.g., Windows®, CDE, HTML, Java)
- Examples: Buttons, menu, lists, icons, windows

GUI Object Representation

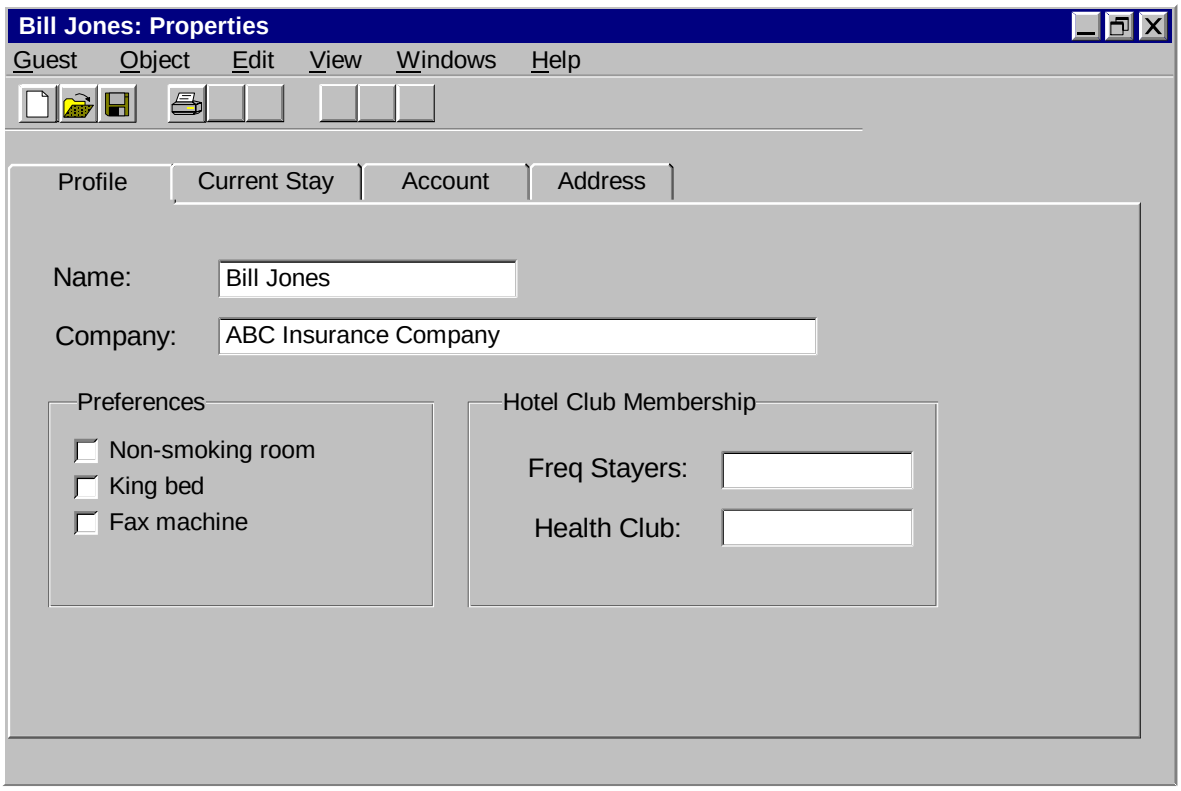
- A UI object may be represented as follows:
  - Closed representation (e.g., list item, icon, menu item)
  - Open representation (e.g., window)



Displaying Object Attributes: Properties

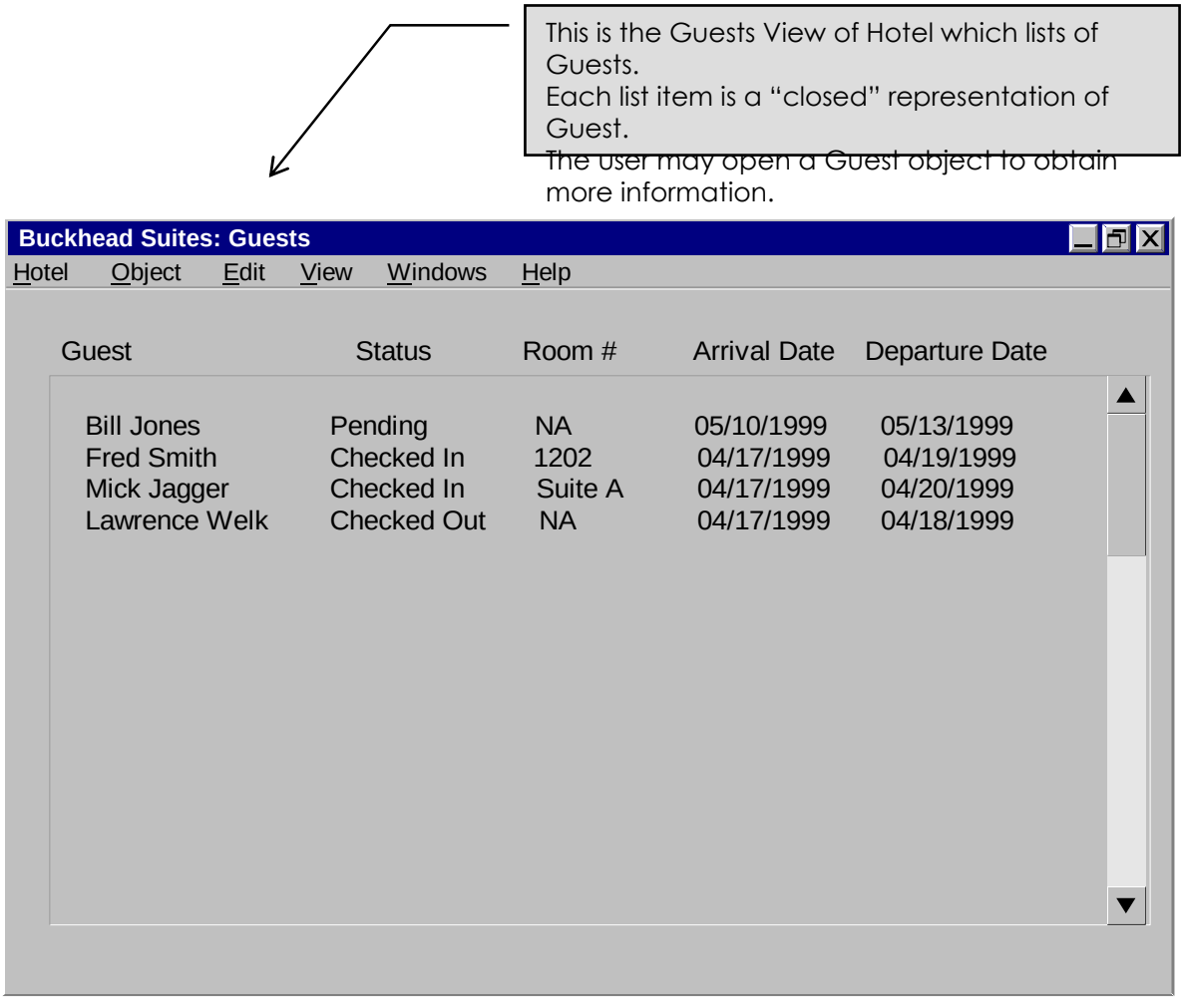
- An Object may have the following types of Attributes:
  - Properties
  - Child Objects

- *Properties* may include the following:
  - Information that describes the Object (e.g., object name, an ID #)
  - Information that describes how the Object is displayed (e.g., default view when opened, color coding within a view, chart style)
  - Information that is contained within the Object, that is not a child object (e.g., Guest Name, Address, Profile)



Displaying Object Attributes: Child Objects

- *Child objects* are contained within the parent object.
  - In the Hotel example, Guests, Rooms, Reservations are child objects of Hotel. All instances of each child object must be represented within the parent object so that the user can manage and manipulate each object instance.
  - A list is used below to display each Guest object instance.
  - Icons may also be used to represent objects.



Primary vs. Secondary Window Types

- *Primary window*
  - Most UI Objects are represented by a Primary Window when opened.
  - Primary windows may be managed independently of all other windows (e.g., closed, minimized, maximized, restored).
    - When a primary window is minimized, an icon is used to represent the closed UI object on the desktop.
  - All primary windows have a menu bar.
    - Common functions found under Edit (e.g., Find, Replace) and View (e.g., Sort) are useful to manipulate the contents of an Object.

- *Secondary window (often referred to as a Dialog Box)*
  - Used to supplement another window (primary or secondary).
    - May be used to obtain or display additional information.
  - A secondary window may only be accessed from its parent window and must close if the parent window is closed.
    - If the parent is minimized, all associated secondary windows are minimized as well.
  - Secondary windows should be modeless unless the task requires user input to continue.
  - A secondary window may be used to display an open object if that object has only 1 view and the pull-down menu functionality is not needed.

Identify the Type of Window to Represent each Object

- *Exercise*: Identify the type of window that should be used to represent each task object:
  - Record the Window Type on each Task Object card. "Window Type: Primary"
  - Create a blank Primary or Secondary window for each object.
    - Create a Title Bar for the window and name the window.
      - > [Object Name] : [View Name]:n
  - Replace the "File" menu label with the name of the object type (e.g., "Guest").

Task Object: **Guest**

Window Type: Primary

Description: A person who has a confirmed reservation within the Hotel.

Attributes:

Name
Billing Address
Confirmation Number

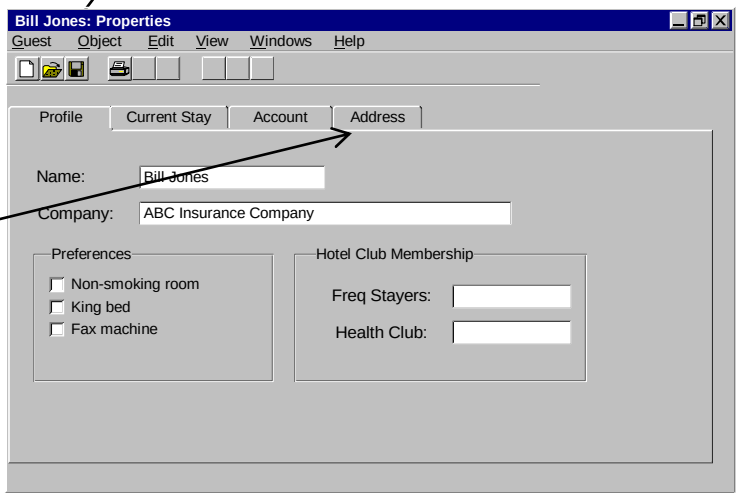
Room

Properties

Actions:
Add, Delete, Edit, Save
Print, Check In, Check Out

I'm In	In Me
Hotel	Room

[Guest- Bill Jones]: [View-Properties]



Views

- When a UI Object is opened, the contents of the Object (child objects and properties) are displayed using one or more Views.
- Every UI Object has at least one View. The View displayed when an Object is opened is the default View. The UI may allow the user to change the default View.
- Different views are used to organize the Object contents in a manner that best support the task/job and in a way that is logical to the user.
- Different views may be used to display different representations of the Object contents (e.g., data view, map view, graphical view).

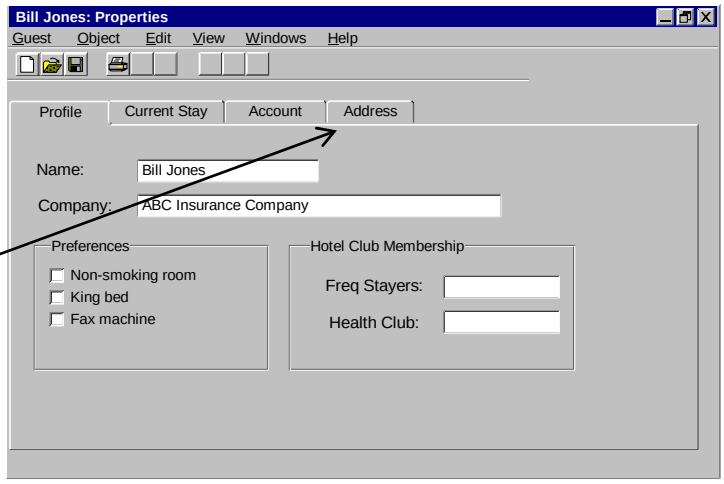
Task Object: **Guest**
Window Type: Primary

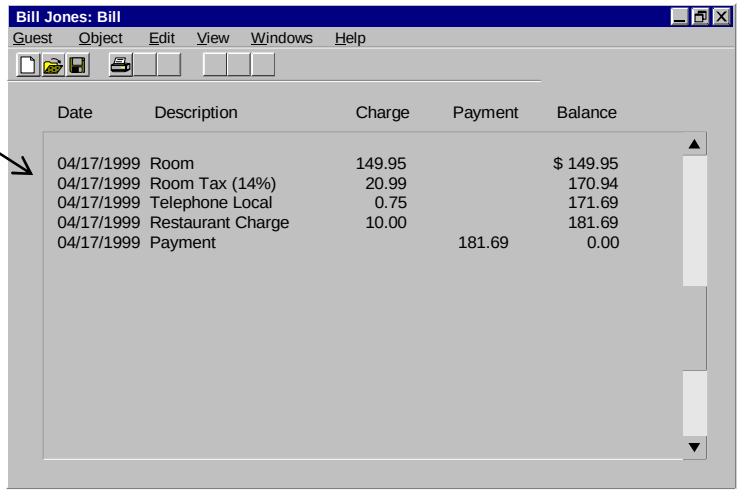
Description: A person who has a confirmed reservation within the Hotel.

Attributes:
Name
Billing Address
Confirmation #
Guest Bill
Room
Properties

Actions:
Add, Delete, Edit, Save
Print, Check In, Check Out

I'm In	In Me
Hotel	Room





Container Views of an Object

- **Views -** Closed representations of an object (e.g., Guest) are typically displayed either as:
  - A list in a separate Container View. In the example below, a Container View of Hotel was built to list the Guest objects.
    - Note that some of the more frequently accessed attributes of Guest are displayed to minimize the need to open the object.
  - A list integrated into another Object View. In many cases, the list of Guest objects may be integrated into another window to minimize switching between views.
    - For example, the default view when the Hotel application starts may contain a list of Guest objects, if viewing Guest properties is the most likely initial task.

Rules for Designing Views

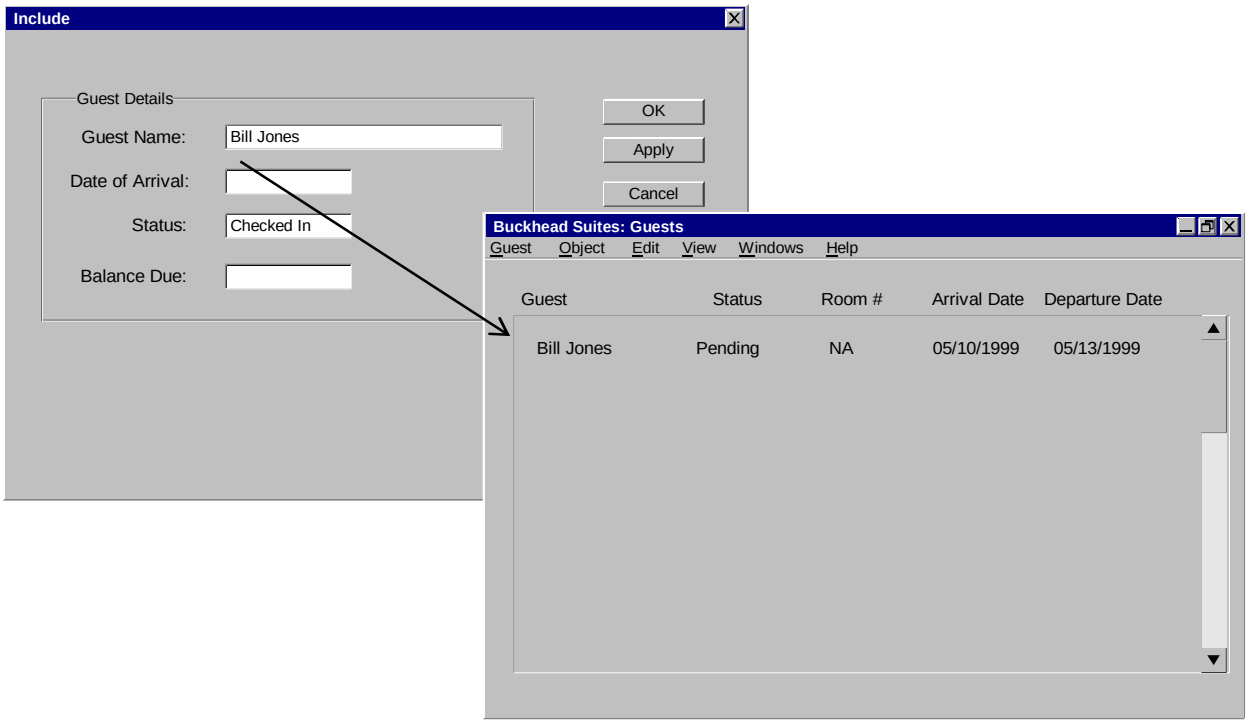
- Each UI Object has at least one view.
- The view that displays when the object is opened is the default view. Allow the user to change the default view in a properties dialog for the View.
- If a UI Object has more than one view::
  - Use a primary window with a pull-down menu bar to represent the object.
    - The pull-down menu bar and its selections are the same across all views of the Object. The menu bar for each different UI Objects *may* vary.
    - Based on the contents of an Object view, some menu selections may be greyed out (unavailable).
    - If a View displays a list of objects (Container View) or properties, provide selections under Edit and View that allow the user to sort, search, and filter the View contents.
  - Allow the user to change the displayed view without opening a new window. Replace the contents of the current view with the new view.
    - Allows the user to minimize the number of open windows.
  - Allow the user to display another view of the Object by opening a new window.
    - Allows the user to view multiple Object views simultaneously.
- Ensure that the user can easily access and open child objects contained within an Object container.
  - For example, provide a Container view that displays a list of all instances of a child object (e.g., Guest, Room) contained within the parent Object (e.g., Hotel).

Include

- The Include function allows the user to limit (filter) the objects displayed and available for selection.
  - The Include function is essential when there may be a large number of Object instances (e.g., Customers, Hotel Guests, Customer orders).

- In our Hotel example, there may be thousands of Guests who either have a reservation or are currently checked into the hotel. Include allows the user to either display the entire list or a sub-set of the list based on certain criteria:
  - For example, date of arrival, guests checked in, guests with an outstanding balance.

- The user specifies the criteria for Include in a secondary window referred to as the Include Dialog. Include criteria may be changed at any time.
  - The user should be able to setup defaults for the Include dialog.
  - Note: Consider the impact on the database when designing your Include dialog.



Sort

- Where a View contains a list of objects, allow the user to Sort the order of the object list (object instances).

- Sort is a common function that enhances the flexibility of your UI design. Sort reduces the need to provide additional object views and gives the user a more powerful tool to perform the task/job.

- Sort may be implemented as…
  - Secondary window under the View pull-down that allows the user to specify sort criteria. This sort criteria may be changed at any time.
  - User clicks on a list heading label to sort by that heading (e.g., Name, Date)

Identify the Views for each Object

- *Exercise*: Identify the Views for each UI Object:
  - Record the View names on each Object index card.
    - For example: Views: Bill Detail, Properties
  - Create a new window for each View.
    - Record the window title on the window. Bill Jones: Bill Detail
  - Note: Don't worry about the window contents yet…

Task Object: **Guest**
Window Type: Primary
Views: Bill Details, Properties

Description: A person who has a confirmed reservation within the Hotel.

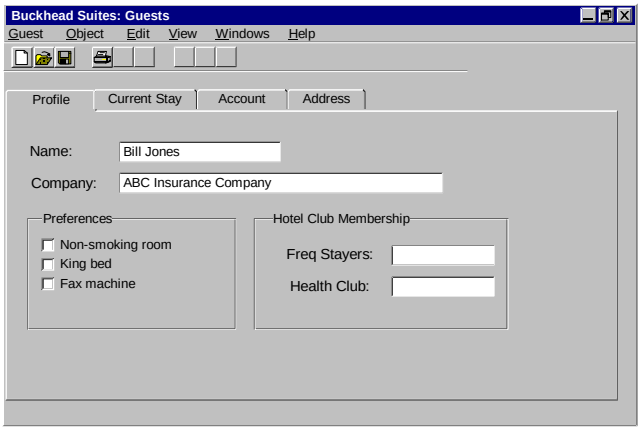
Attributes:
Name Billing Address
Confirmation #
Bill Detail

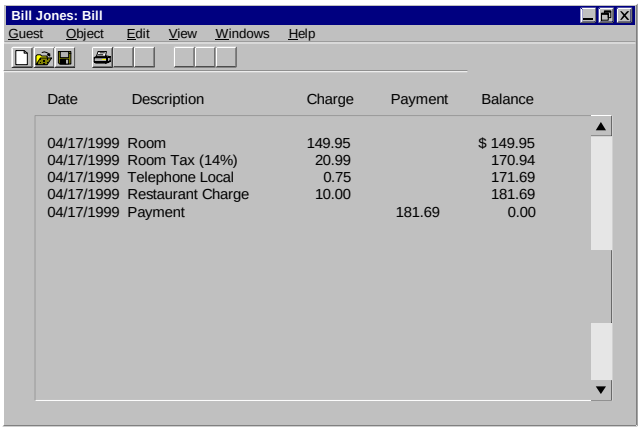
Room

Properties

Actions:
Add, Delete, Edit, Save
Print, Check In, Check Out

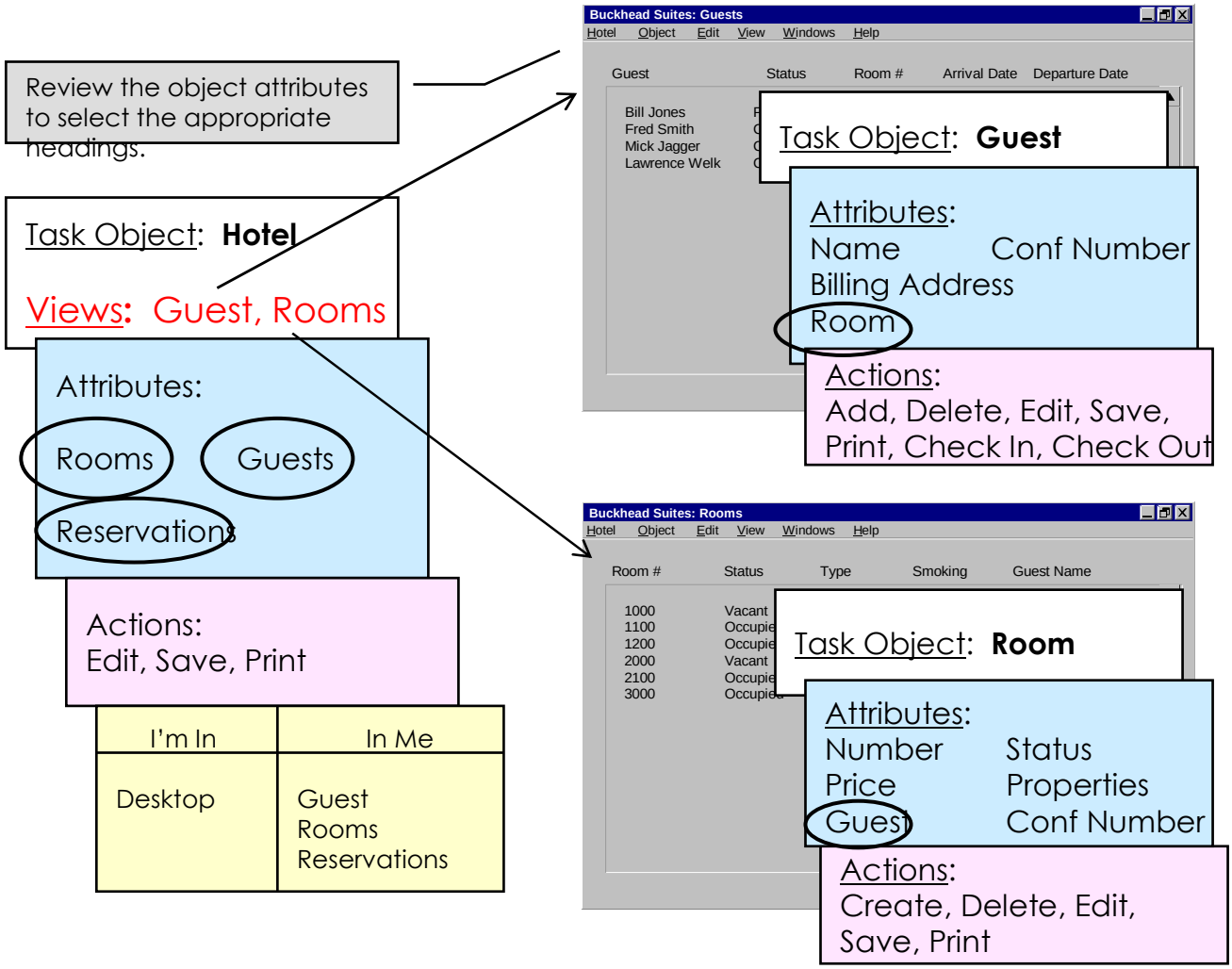
I'm In	In Me
Hotel	Room





Identify the Views of each Object (Cont'd)

- *Exercise Cont'd*: Some of the Objects will have Container Views. These are views that typically contain lists of child objects that may be opened.
  - Be sure that all View names are recorded on each Object index card.
    - For example: Views: Guest List, Rooms
  - Be sure to create a View window for each of these container views.
    - Record the window title on the window. Buckhead Suites: Guest List
  - From the child object's (Guest, Room) attribute list, select the attributes that are most frequently needed by the user. Use these attributes as column headings in the container view.



Verify Usability and Completeness of the UI Design

- *Goal*:
  - Ensure that the set of UI Object windows allow the user(s) to easily and efficiently perform the task flows defined in Part 1.
    - Usability must be acceptable to the user(s).
  - Fix any UI Object windows, Task Objects, or Task Flows found to be incorrect or incomplete.
    - Since this is an iterative design process, we will continue to make changes and verify our design throughout the workshop.
    - Changes to any windows, task objects, or task flows should be verified.
    - Continue the iterative process of the test, fix, retest until the team agrees to move on.

- *Exercise:*: Verify the usability, accuracy, and completeness of the UI object windows.
  - Represent the product/application on the desktop (e.g., desktop icon).
  - A team member (other than the user) should select a task from the Task Flow layout and read through each step.
  - As each step is read, a user representative on the team should attempt to use the UI windows to execute the task. The user should describe what would happen in each window.
    - Fix any windows, task objects, or the task flow as needed.
    - Retest any changes.
  - All other team members should watch the task execution and help verify the design.
  - Be sure that all tasks are attempted using the UI design.
  - Continue the iterative process of the test, fix, retest until the team agrees to move on.

When to use Menus and Command Buttons

- Menu Bars
  - Use a menu bar in windows that:
    - ⌫require selections found in the *File, Object, Edit, View,* and *Window* pull-down menus.
      - > Primary windows should be implemented with a menu bar.
    - ⌫have more than 6 command buttons (rule of thumb).
      - > Implement the command buttons as pull-down menu selections to avoid cluttering the window with buttons.

- Command Buttons
  - Use command buttons in windows that:
    - ⌫do *not* require selections found in the *File, Object, Edit, View,* and *Window* pull-down menus.
    - ⌫Transaction and message dialogs that are typically displayed for a short period of time are good examples of windows that use command buttons.

Pull-down Menus

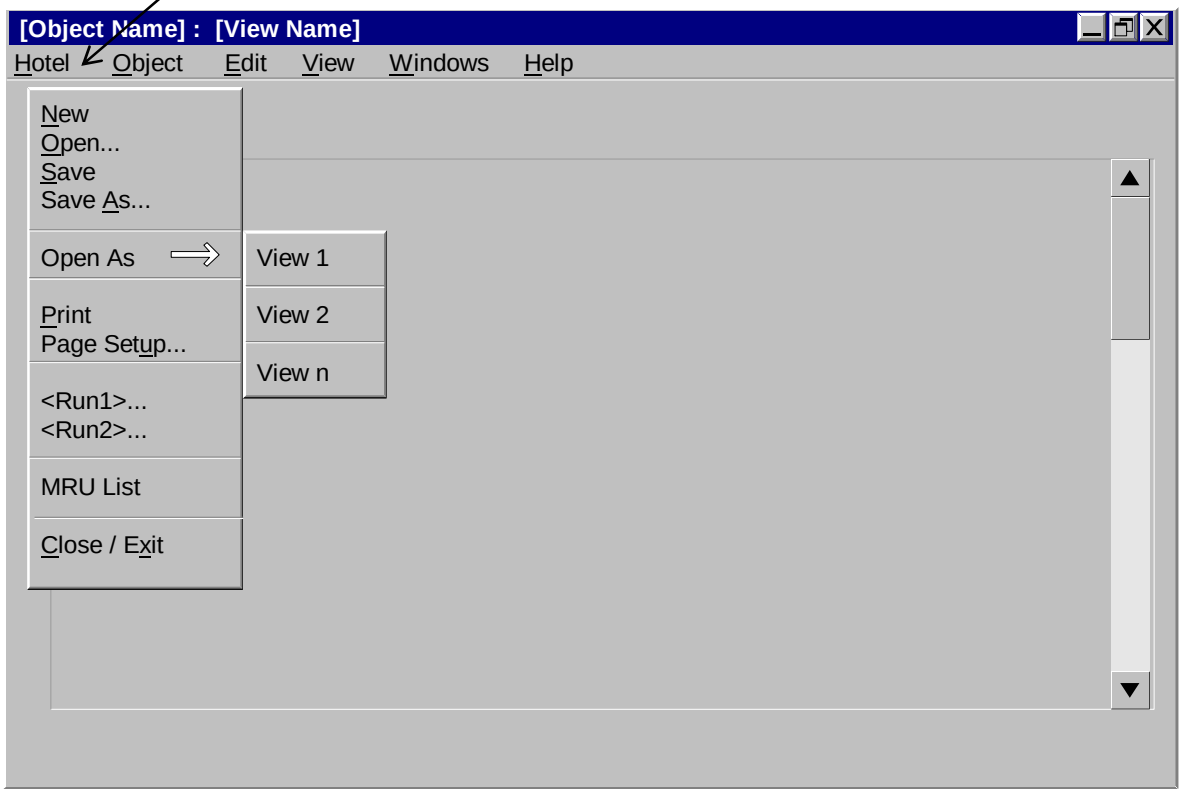
- Each application/product designed using the UI Framework provided with this workshop will implement a consistent set of pull-down menus, customized for the application/product.
- Each pull-down menu has an associated scope. Thus, a user selects a pull-down menu based on the object to be manipulated or managed.
- It is important to understand the scope of each pull-down menu within the UI Framework used in this workshop.

  - File <*UI object name*> - Scope is the primary window and all child windows that represents the UI object name.
  - Object (Windows™) / Selected (Motif™/CDE) or *Object*). Scope is the UI objects represented in the open window (e.g., list item) that may be selected.
    - New <object name> is an exception. This selection may be used to create a new instance of any object class represented in the UI.
  - Edit - Scope is the content of the open window (properties and child UI objects).
  - View - Scope is the visible contents of the open window. Functions in View change the contents (e.g., Include, Change View) of the window or the way the contents are displayed (e.g., Sort).
  - Windows - Scope is all windows that are part of the UI design. Windows contains window management functions.

- For each UI object, the pull-down menu structure of each View contains the same menus and selections.
  - If a menu selection does not apply or is not available in a View, disable the selection. Do not remove the selection from the menu.

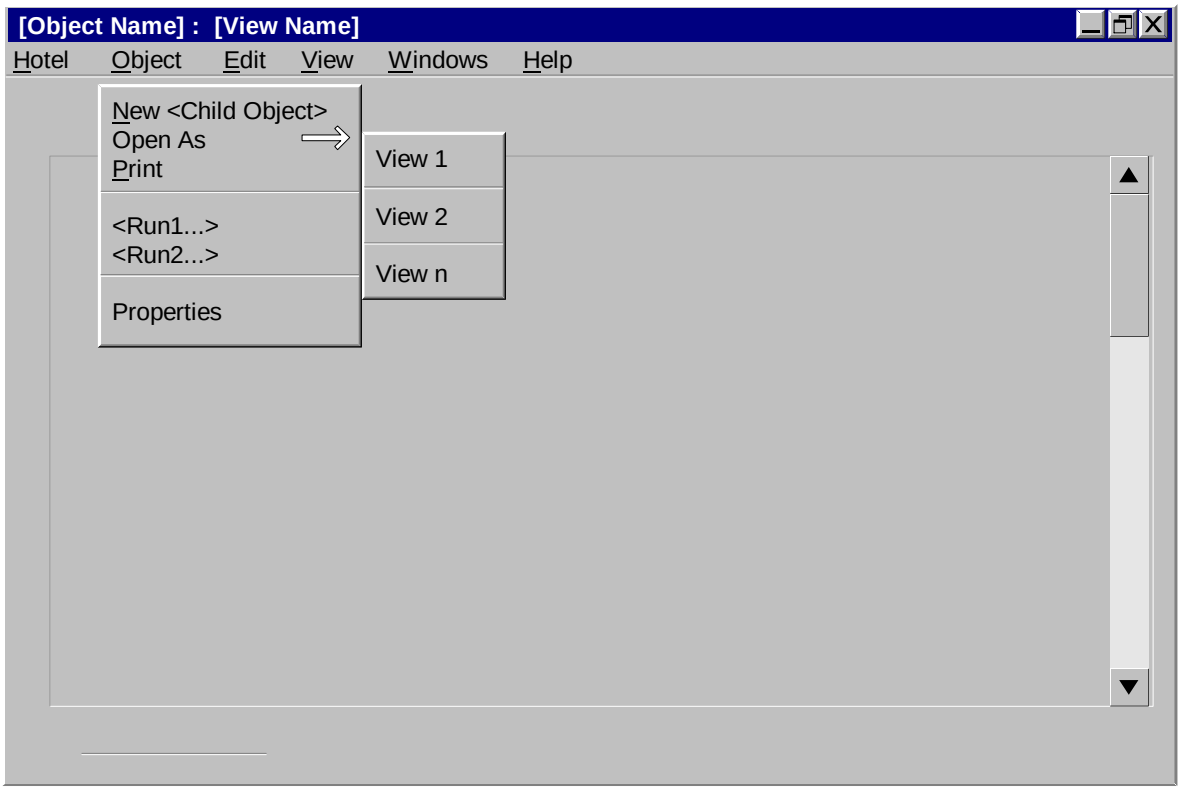
File <Object Name> Pull-down Menu

- The File pull-down menu applies to the primary window and all child windows that represents the UI object name
- File may be renamed to be the name of the object's class represented by the primary window.
  - For example, Hotel is the name of the Task Object, Hotel, that is the parent container window for the Hotel example application.



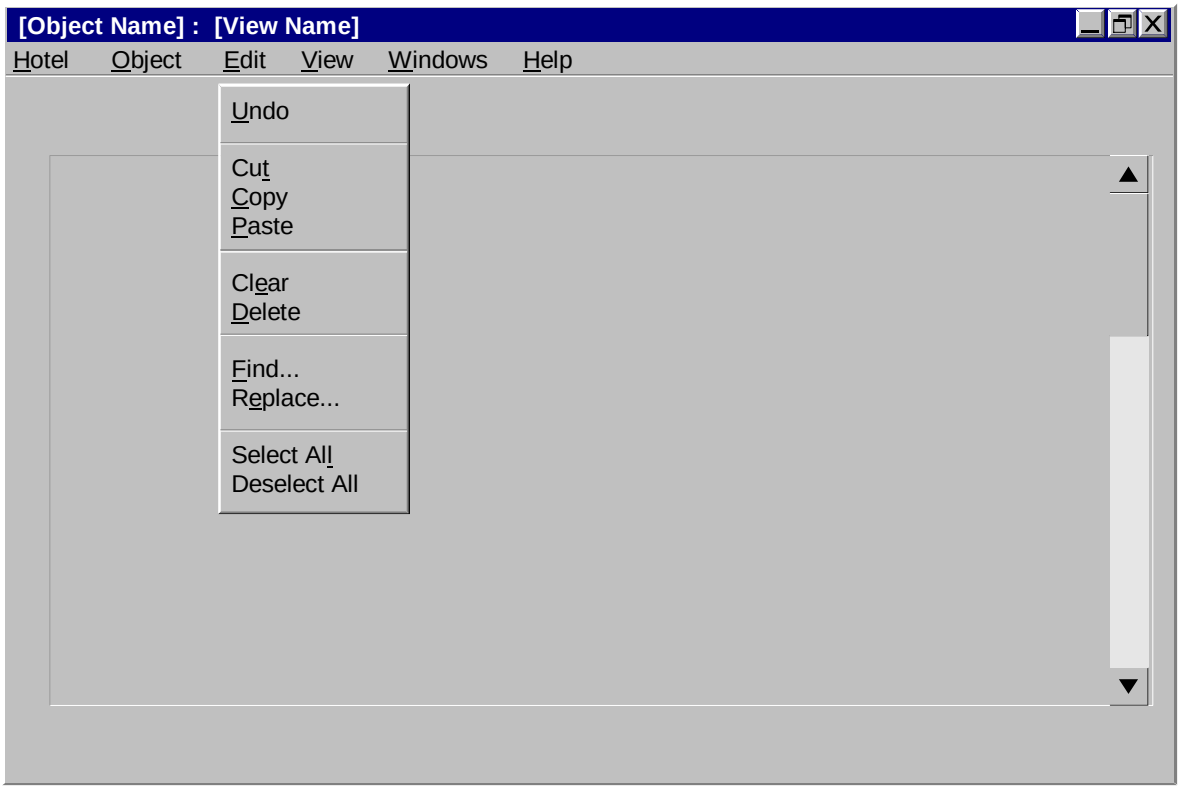
Object/Selected Pull-down Menu

- The Object/Selected pull-down menu affects the UI objects represented in the open window (e.g., list item) that may be selected.
- The Objects menu only affects objects that are visible in the open view.
- The New <child object> selection is an exception. It allows the user to create a new instance of any UI object class represented in the UI.
- This pull-down is label Object in Windows™ applications and Selected in Motif/CDE applications.



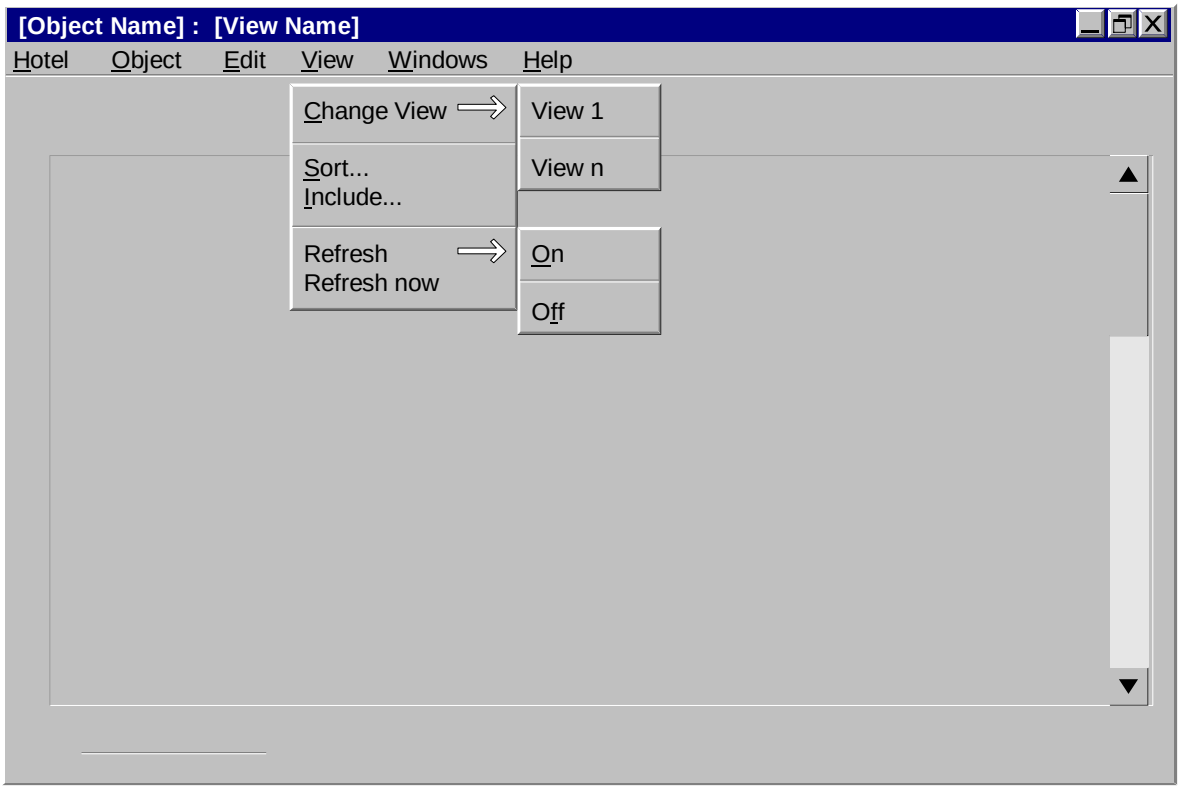
Edit Pull-down Menu

- The Edit pull-down menu affects the content of the open window (properties and child UI objects).



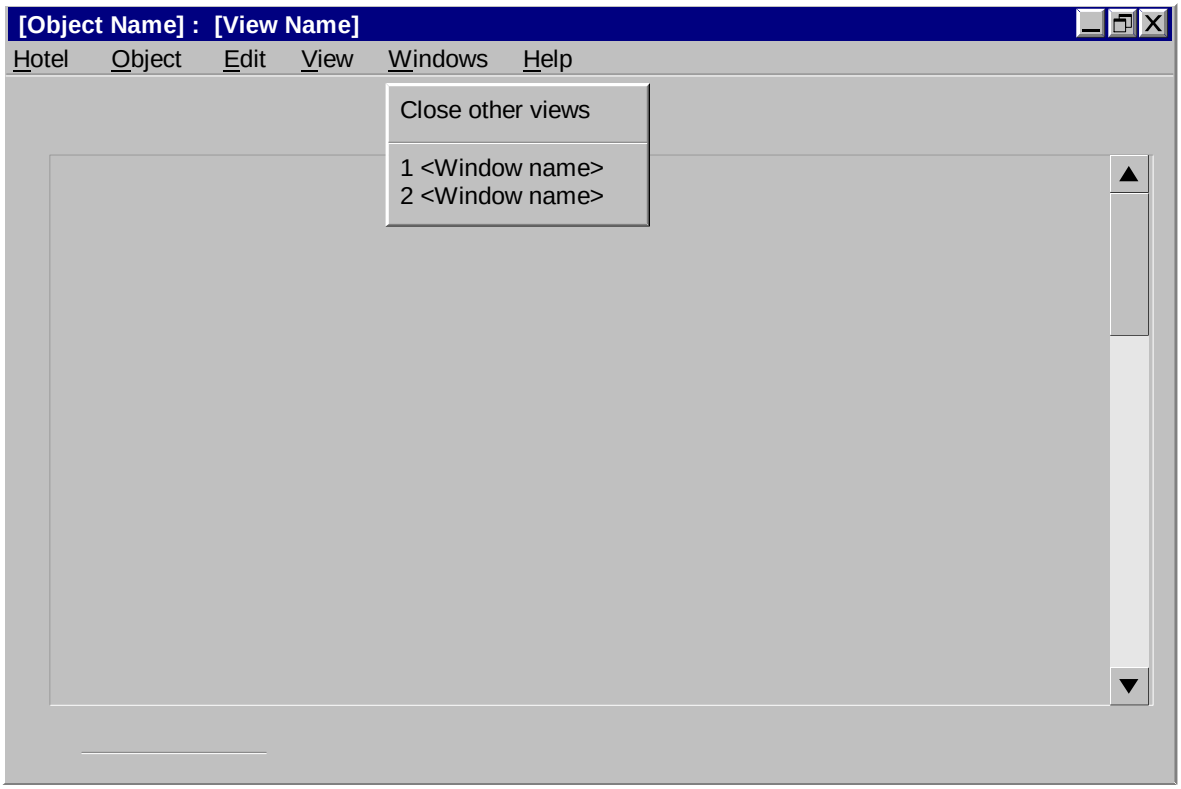
View Pull-down Menu

↗ The View pull-down menu affects the the visible content of the open window.



Windows Pull-down Menu

↗ The Windows pull-down menu affects all windows that are part of the UI design. Windows contains window management functions



Command Buttons

- Command buttons invoke commands/actions that affect the entire secondary window (dialog).
  - Buttons that invoke commands/actions on a sub-set of data within the secondary window are referred to as Client buttons and are not covered in this topic.

- Below is a list of commonly used command buttons in dialog windows:
  - Properties and transaction dialogs:
    - OK
    - Apply
    - Reset
    - Cancel
    - Help
  - Process dialogs:
    - Retry
    - Stop
    - Continue
    - Pause
    - Resume
    - Cancel
    - Help
  - Message dialogs:
    - Yes
    - No
    - OK
    - Cancel
    - Help

Design the Pull-down Menus and Command Buttons for each Object Window

↗ Exercise: Design the pull-down menus and command buttons for each UI object.

- For each UI Object represented by a primary window, design the pull-down menus:
  - ⌧Design the File, Object, Edit, View, and Window pull-down menus. For each menu pull-down:
    - > Use a separate sticky for each pull-down menu.
    - > Using the pull-downs described in the previous slides, determine which selections are appropriate to support the tasks and commands for the UI object.
    - > Be sure to include the actions listed on the pink Action sticky.
    - > Record all the menu selections on the respective sticky.
  - ⌧Attach the pull-down menu stickies to a blank sheet of paper.
    - > Record the UI object name on the sheet of paper.
  - ⌧Be sure to review the actions for each view of the UI Object.

- For each UI Object represented by a secondary window containing command buttons:
  - ⌧Design the command buttons.
  - ⌧Label each button with the appropriate action (e.g., Ok, Close, Help)
  - ⌧Use a separate sticky for each button.
  - ⌧Attach the stickies onto the secondary window.

Verify Usability and Completeness of the UI Design

- *Goal*:
  - Ensure that the set of UI Object windows allow the user(s) to easily and efficiently perform the task flows defined in Part 1.
    - Usability must be acceptable to the user(s).
  - Fix any UI Object windows, Task Objects, or Task Flows found to be incorrect or incomplete.
    - Since this is an iterative design process, we will continue to make changes and verify our design throughout the workshop.
    - Changes to any windows, task objects, or task flows should be verified.
    - Continue the iterative process of the test, fix, retest until the team agrees to move on.

- *Exercise:*: Verify the usability, accuracy, and completeness of the UI object windows.
  - Represent the product/application on the desktop (e.g., desktop icon).
  - A team member (other than the user) should select a task from the Task Flow layout and read through each step.
  - As each step is read, a user representative on the team should attempt to use the UI windows to execute the task. The user should describe what would happen in each window.
    - Fix any windows, task objects, or the task flow as needed.
    - Retest any changes.
  - All other team members should watch the task execution and help verify the design.
  - Be sure that all tasks are attempted using the UI design.
  - Continue the iterative process of the test, fix, retest until the team agrees to move on.