An Introduction to Object-Oriented Analysis and Design

PART VII: Application

Dr. 邱 明
Software School of Xiamen University
mingqiu@xmu.edu.cn
Fall, 2009

Chapter 12: Web Application: Vacation Tracking System

Object-Oriented Analysis and Design with Applications (Third Edition)
Grady Booch, Robert A. Maksimchuk, Michael W. Engle, etc.
Posts & Telecom Press
2008.4
978-7-115-17305-5/TP

12.1 Inception

- **The Requirements**
  - A summary of the vision document,
  - Key features,
  - The use case model,
  - Key use case specifications,
  - Architecturally significant line item requirements.

12.1 Inception

w The vision document

A Vacation Tracking System (VTS) will provide individual employees with the capability to manage their own vacation time, sick leave, and personal time off, without having to be an expert in company policy or the local facility's leave policies.

12.1 Inception

- The most important goal
  - Give individual employees the capability and responsibility to manage their vacation time
  - The need to streamline the functions of the human resources (HR) department
    - Minimize noncore, business-related activities of management
    - Give a sense of empowerment to the employees
  - Improve the internal business processes of managing vacation time requests

The system must be easy to use.

12.1 Inception

- The conventional way,
  - All vacation time had to be approved by an immediate manager
  - Then checked by a clerk in the HR department before it was authorized.
  - Sometimes this manual process could take days.
- The VTS
  - Will require at most one manual approval by the immediate manager
  - Save time and money

12.1 Inception

- The system will provide the following key features:
  - Implements a flexible rules-based system for validating and verifying leave time requests
  - Enables manager approval (optional)
  - Provides access to requests for the previous calendar year, and allows requests to be made up to a year and a half in the future
  - Uses e-mail notification to request manager approval and notify employees of request status changes
  - Uses existing hardware and middleware
  - Is implemented as an extension to the existing intranet portal system, and uses the portal's single-sign-on mechanisms for all authentication

12.1 Inception

- Keeps activity logs for all transactions
- Enables the HR and system administration personnel to override all actions restricted by rules, with logging of those overrides
- Allows managers to directly award personal leave time (with system-set limits)
- Provides a Web service interface for other internal systems to query any given employee's vacation request summary
- Interfaces with the HR department legacy systems to retrieve required employee information and changes

12.1 Inception

- **The Use Case Model**
- The system contains the following actors.
  - Employee: The main user of this system.
    - uses this system to manage his or her vacation time.
  - Manager:
    - has all the abilities and goals of a regular employee,
    - with the added responsibility of approving vacation requests for immediate subordinates.
    - may award subordinates comp time, subject to certain limits set in the system.

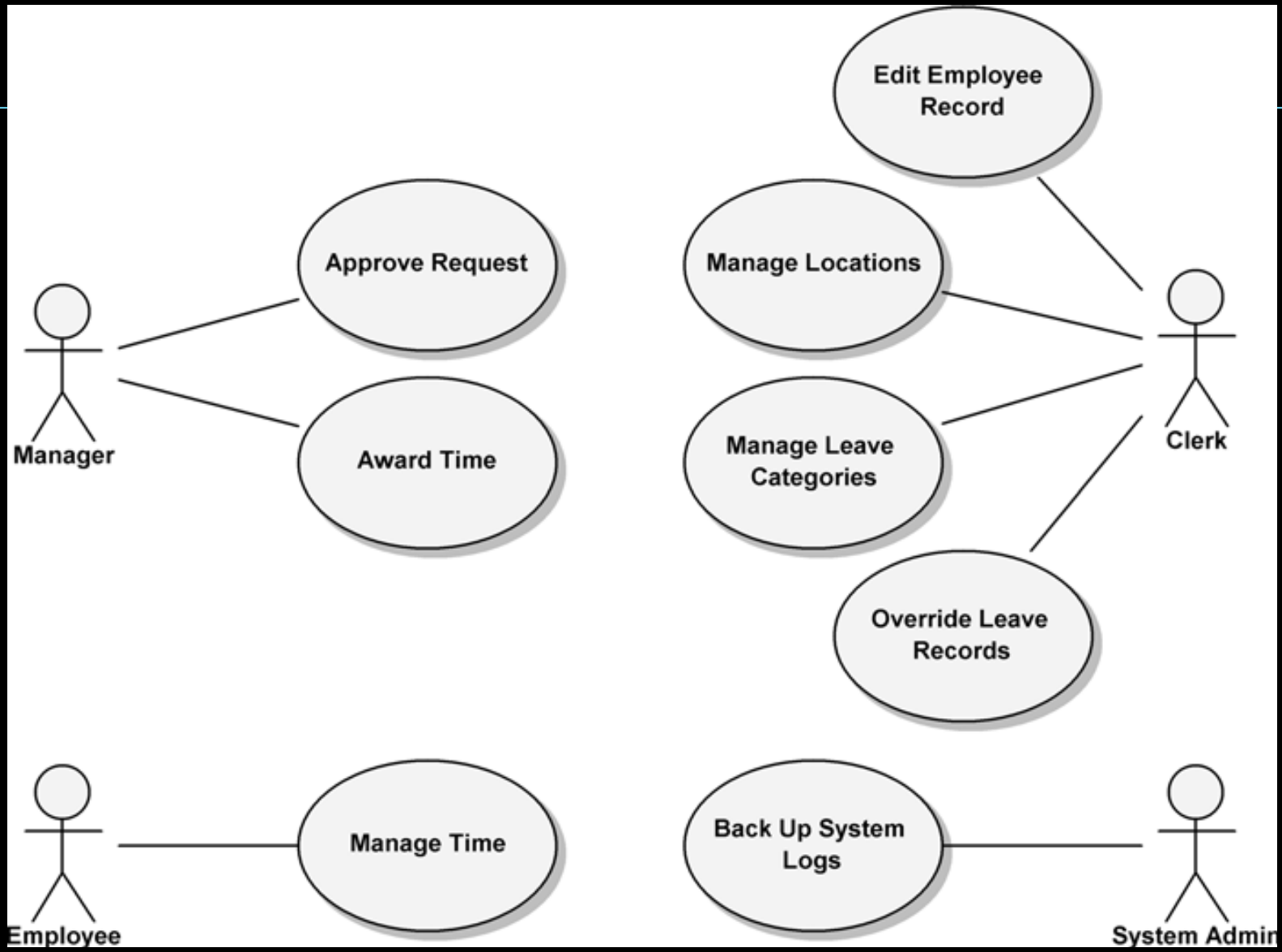
12.1 Inception

§ Clerk:

- has sufficient rights to view employees' personal data
- responsible for ensuring that employees' information is up to date and correct.
  - w can add or remove nearly any record in the system.
- In the real world, HR clerks may or may not be employees;
  - w however, if they are employees, they use two separate login IDs to manage these two different roles.

§ System Admin:

- A role responsible for the smooth running of the system's technical resources (e.g., Web server, database) and for collecting and archiving all log files.



A use case is not a description of a single functional requirement but rather a description of something that provides significant value to the actor invoking it, in the form of scenarios

12.1 Inception

- The main use cases are as follows.
  - Manage Time:
    - Describes how employees request and view vacation time requests.
  - Approve Request:
    - Describes how a manager responds to a subordinate's request for vacation time.
  - Award Time:
    - Describes how a manager can award a subordinate extra leave time (comp time).
  - Edit Employee Record:
    - Describes how an HR clerk edits an employee's information in the system.
      - includes setting all the leave time allowances and the maximum time that can be awarded by the manager.

12.1 Inception

§ Manage Locations:
- Describes how an HR clerk manages location records and their rules.

§ Manage Leave Categories:
- Describes how an HR clerk manages leave categories and their rules.

§ Override Leave Records:
- Describes how an HR clerk may override any rejection of leave time requests made by the rules in the system.

§ Back Up System Logs:
- Describes how the system administrator backs up the system’s logs.

12.2 Elaboration

- In the ideal world,
  - analysis of a system should be independent of an implementing architecture.
- The reality, however, is that
  - prior knowledge of the implementing architecture may contribute to the shape of the analysis.

12.2 Elaboration

- For example
  - The statement in the main flow of the Manage Time use case

> “The employee should have access to a visual calendar to help select and compare selected dates,”

12.2 Elaboration

- 4+1 architecture view model
  - 4 views are:
    - Logical view
    - Process view
    - Deployment view
    - Implementation view
  - +1 view is
    - Use case view

12.2 Elaboration

- Logical view
  - Conceptual organization of the software in terms of the most important layers, subsystems, packages, frameworks, classes, and interfaces.
  - Summarizes the functionality of the major software elements, such as each subsystem.
  - Shows outstanding use-case realization scenarios (as interaction diagrams) that illustrate key aspects of the system.
  - A view onto the UP Design Model, visualized with UML package, class, and interaction diagrams

12.2 Elaboration

- Process View
  - Processes and threads. Their responsibilities, collaborations, and the allocation of logical elements (layers, subsystems, classes, …) to them.
  - A view onto the UP Design Model, visualized with UML class and interaction diagrams, using the UML process and thread notation.

12.2 Elaboration

- Deployment View
  - Physical deployment of processes and components to processing nodes, and the physical network configuration between nodes.
  - A view onto the UP Deployment Model, visualized with UML deployment diagrams.

12.2 Elaboration

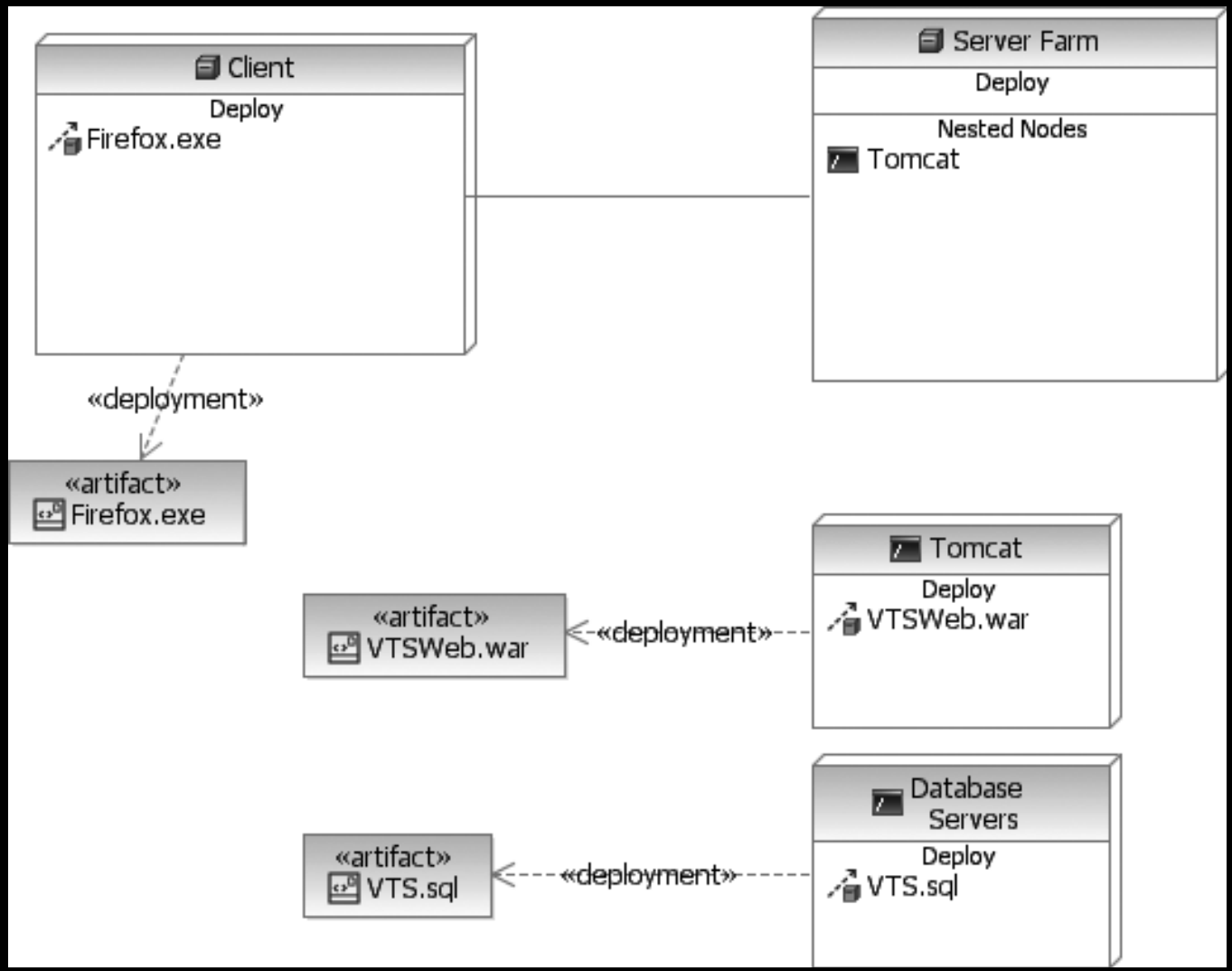
- Implementation View
  - The actual source code, executables, and so forth. It has two parts:
    - deliverables,
    - things that create deliverables (such as source code and graphics).
  - A view onto the UP Implementation Model, expressed in text and visualized with UML package and component diagrams.

12.2 Elaboration

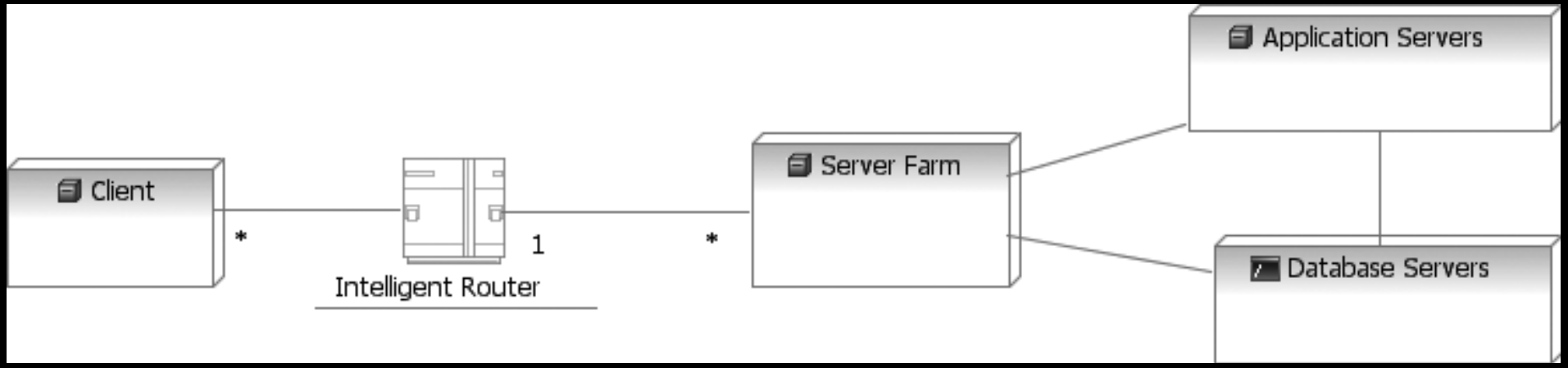
- Use case
  - Summary of the most architecturally significant use cases and their non-functional requirements.
  - A view onto the UP Use-Case Model, expressed in text and visualized with UML use case diagrams and perhaps with use-case realizations in UML interaction diagrams.

12.2 Elaboration

w The Deployment View



12.2 Elaboration



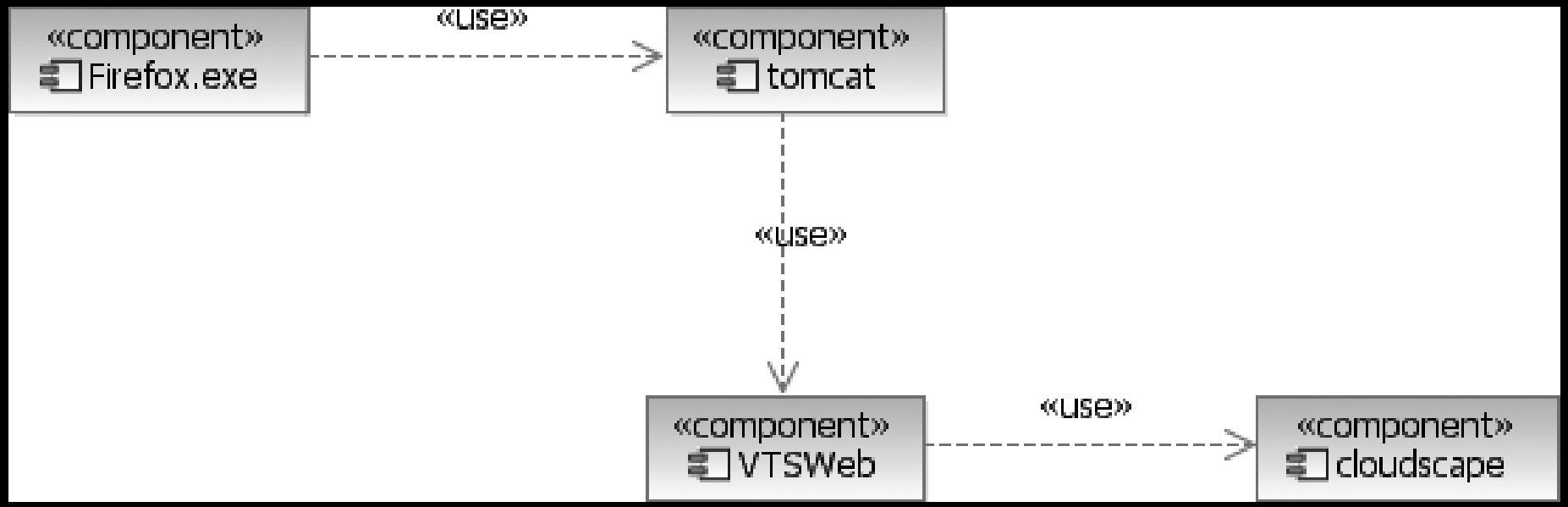
A Deployment Diagram for a Web Application Server Farm with Parallel and Redundant Processing Nodes

12.2 Elaboration

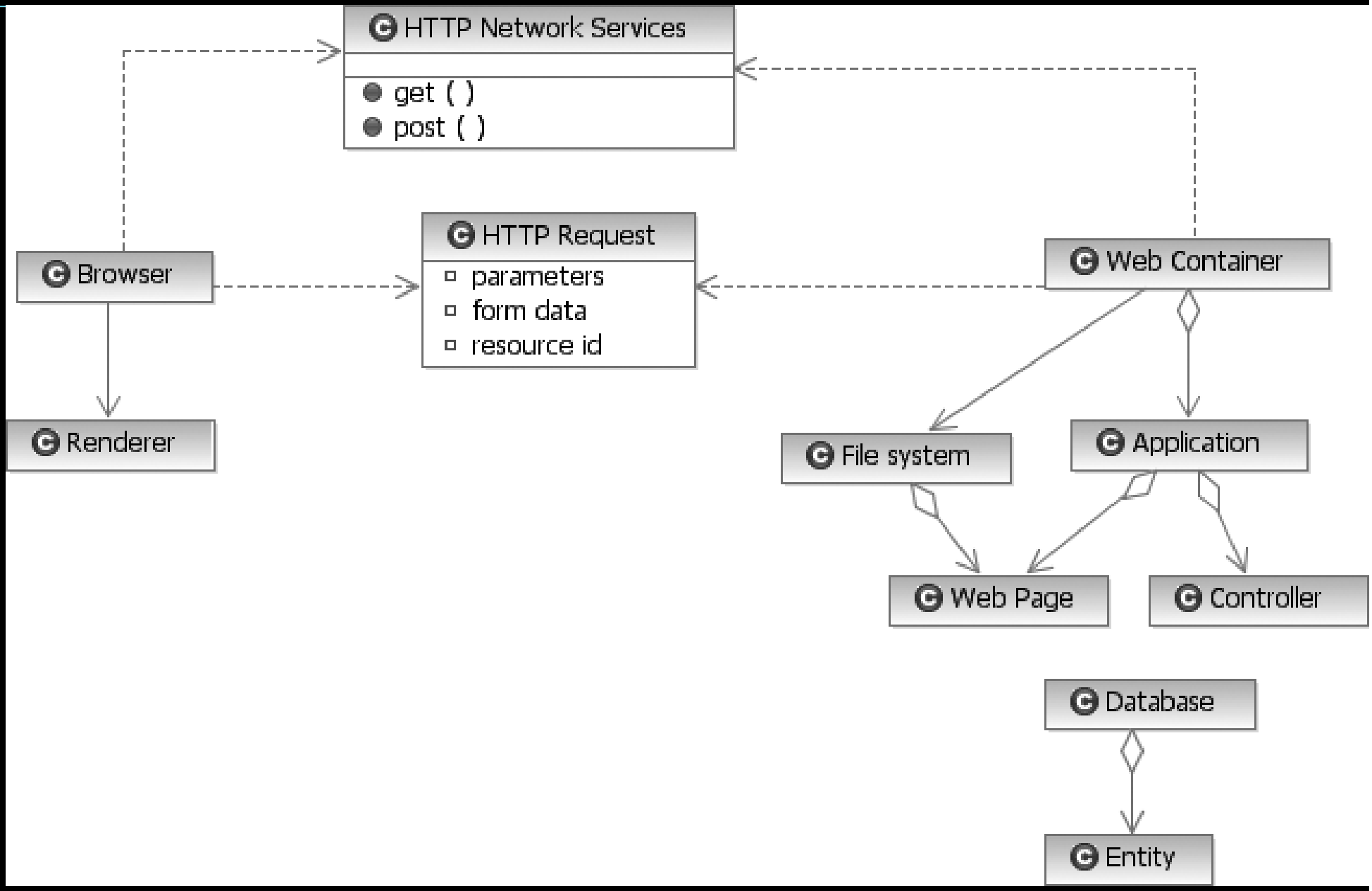
The Logical View

- The typical Web application has at least four logical components:
  - the browser running on the client
  - the Web or application container
  - a separate component for the application logic itself
  - a database server component.

12.2 Elaboration

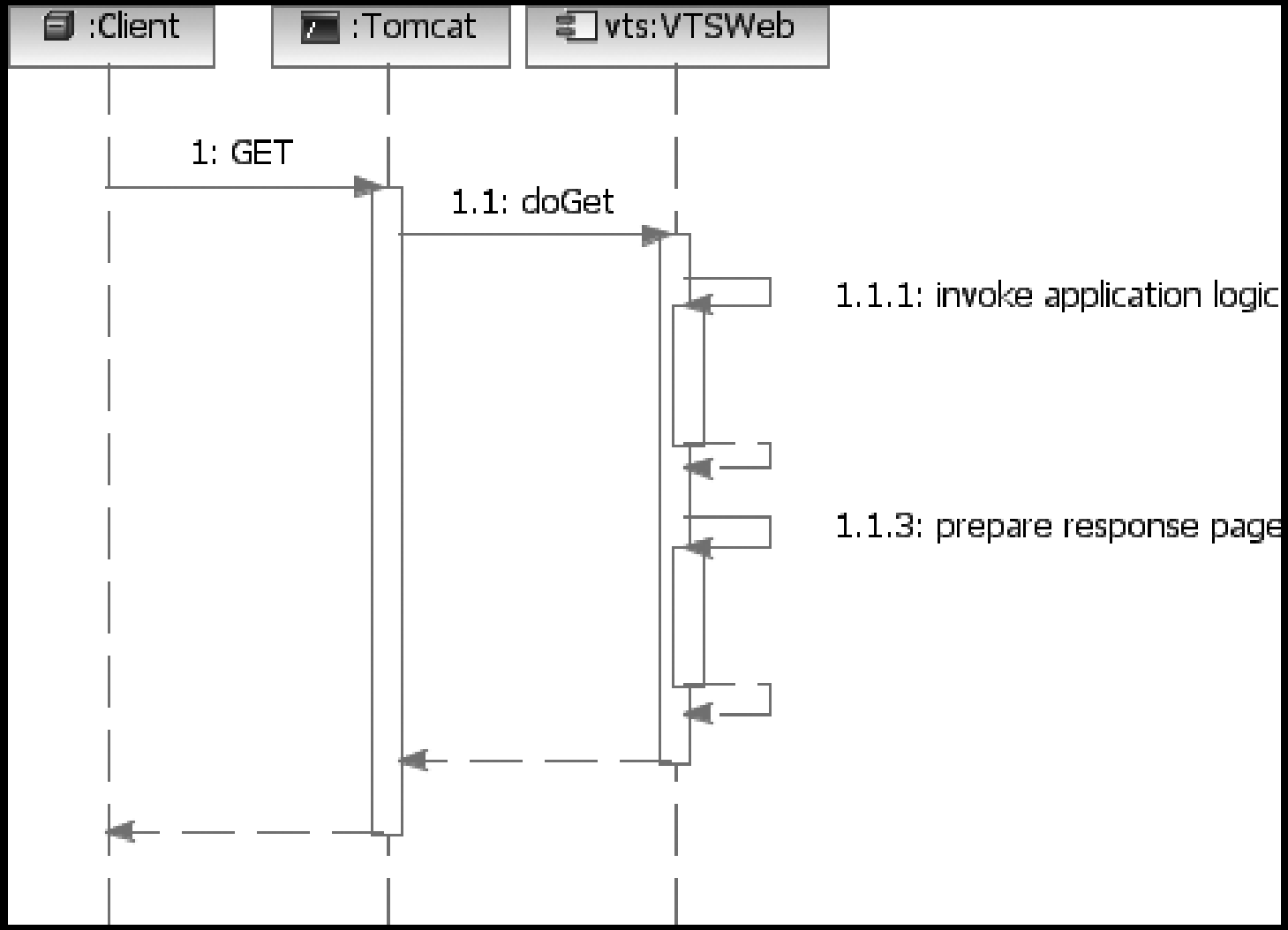


The Logical View of the Primary Components of a Web Application

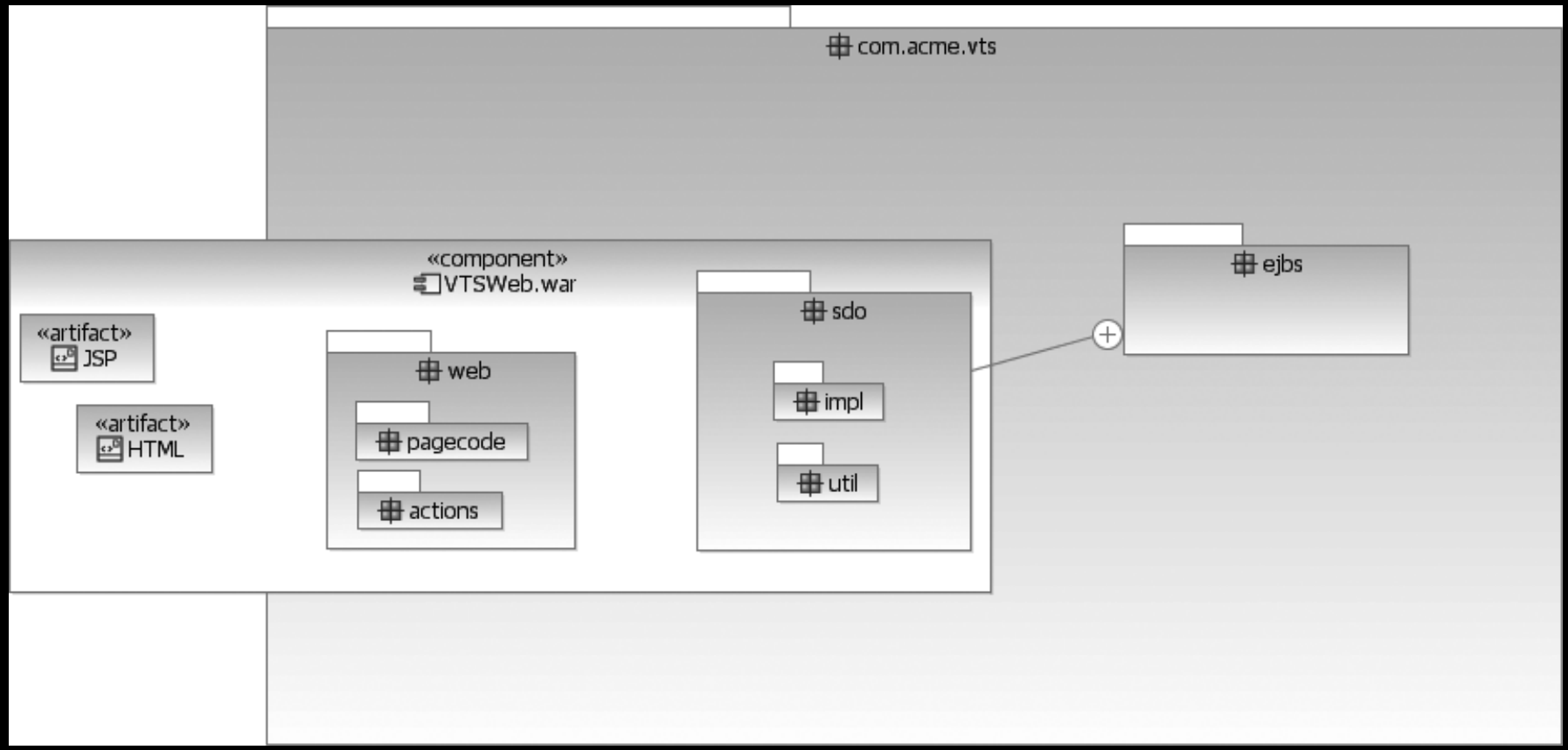

HTTP Network Services
get ( )
post ( )
HTTP Request
parameters
form data
resource id
Browser
Renderer
Web Container
File system
Application
Web Page
Controller
Database
Entity

12.2 Elaboration

w The Process View



w The Implementation View



The Implementation Model Overview

12.2 Elaboration

The Use Case View

- The architecturally representative use case

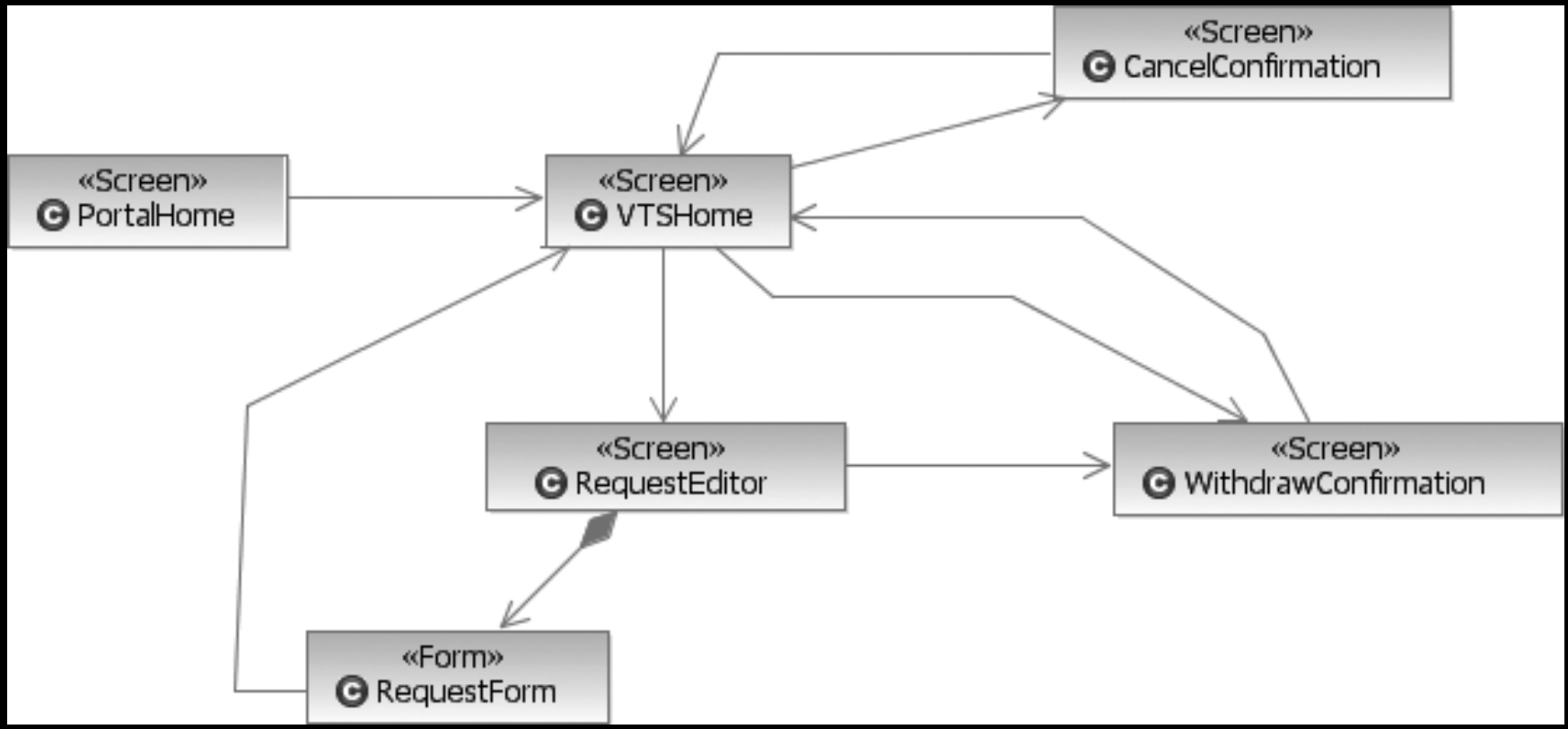
Manage Time

  - the most frequently invoked
  - most viewed by all the actors of the system
- Use case
  - a functional description of the system from the viewpoint of the Employee actor.

12.2 Elaboration

- Domain knowledge Examples
  - All employees work eight-hour days.
  - Each employee’s vacation time requests are subject to the restrictions of each employee’s primary work location in addition to overall company policies and restrictions.
  - Vacation time request validation rules are defined and owned by the HR department.

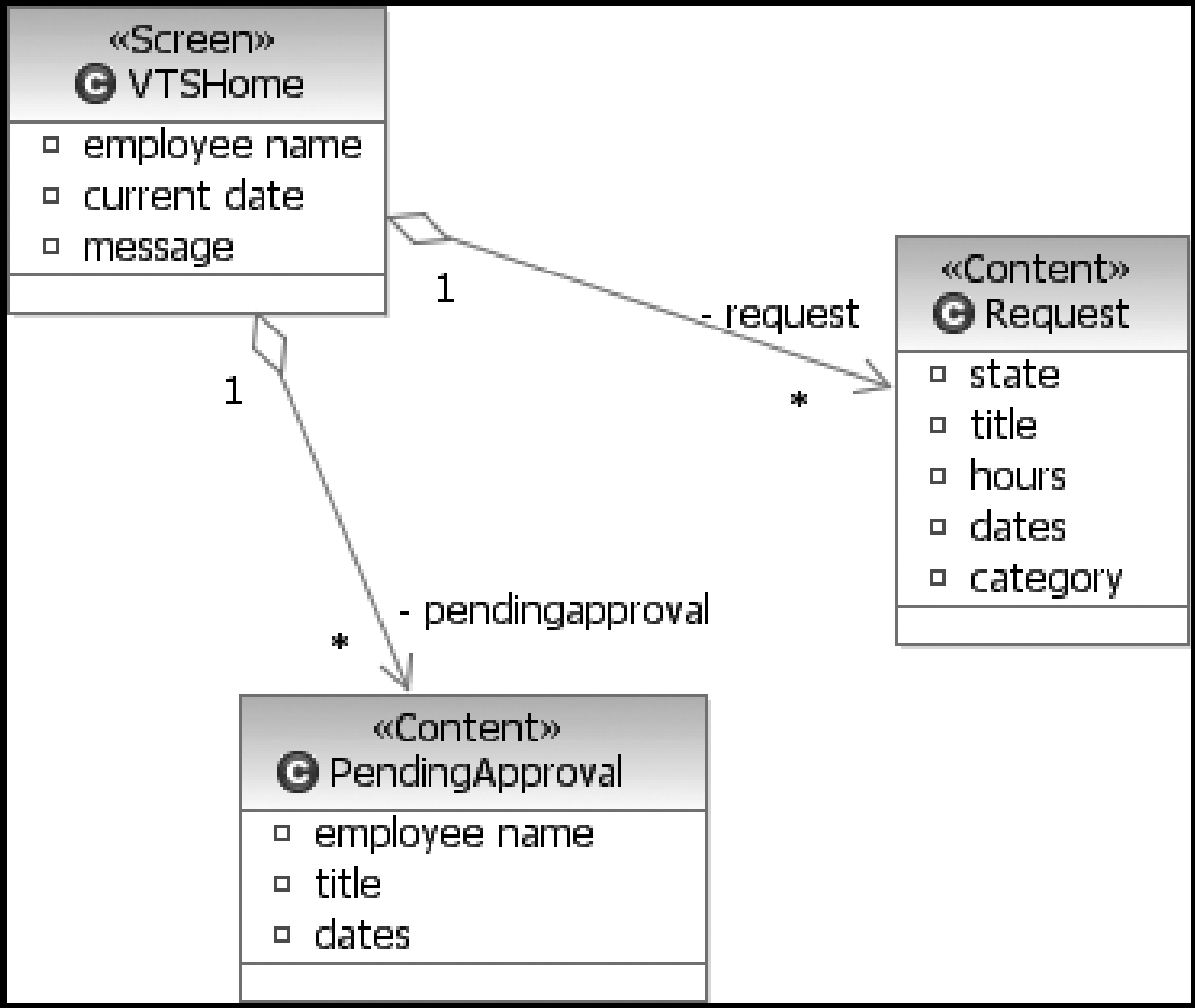
w The User Experience Model



A High-Level User Experience Model

12.3 Construction

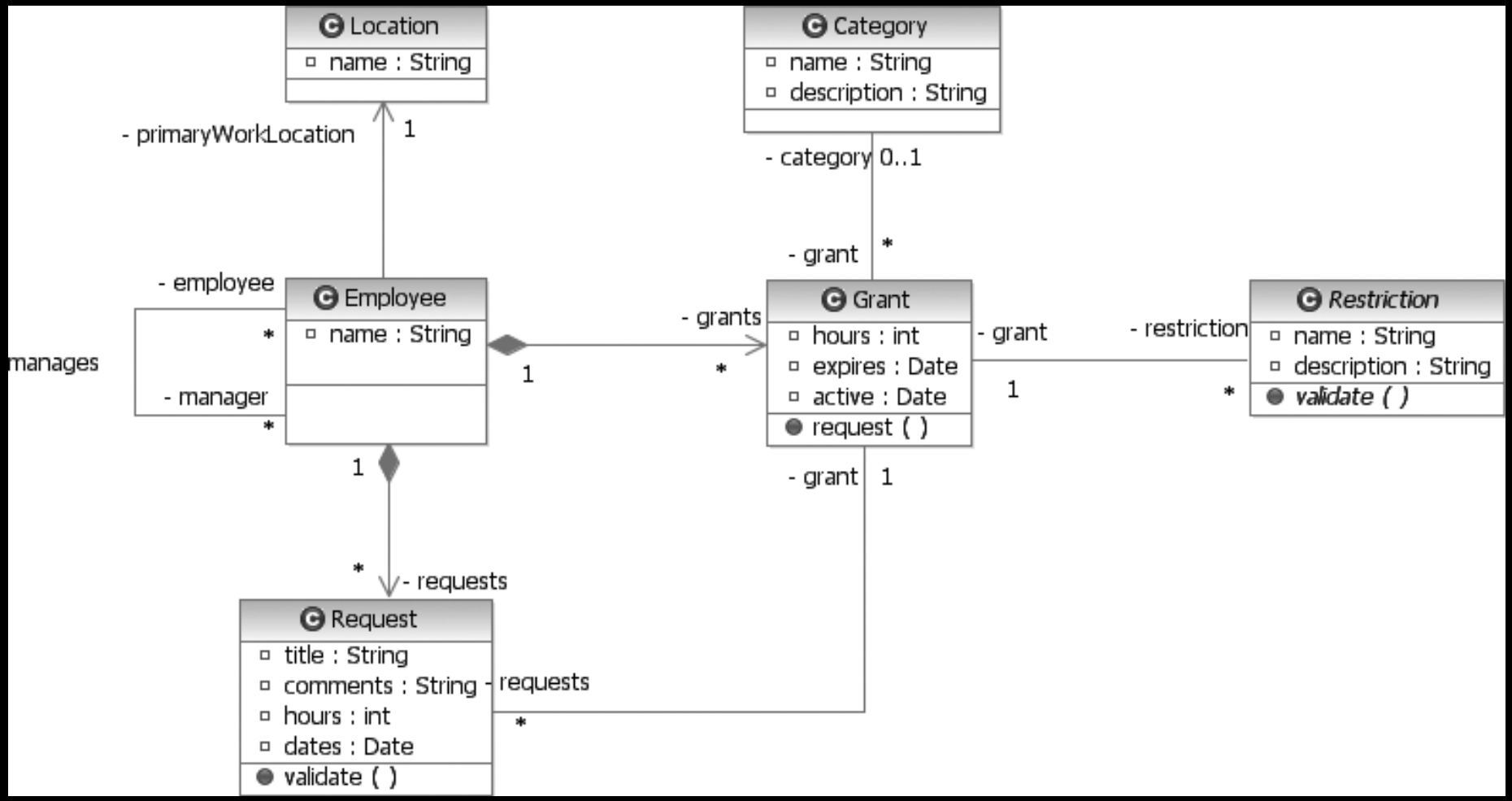
A Detailed View of the VTSHome Class



12.3 Construction

- **The Analysis and Design Models**
  - The Analysis Model
    - Capture the entities processes of the system
    - The first attempt at identifying the elements that will make up the solution space
      - do a simple noun-verb analysis of the use case specifications and related requirements documents

12.3 Construction



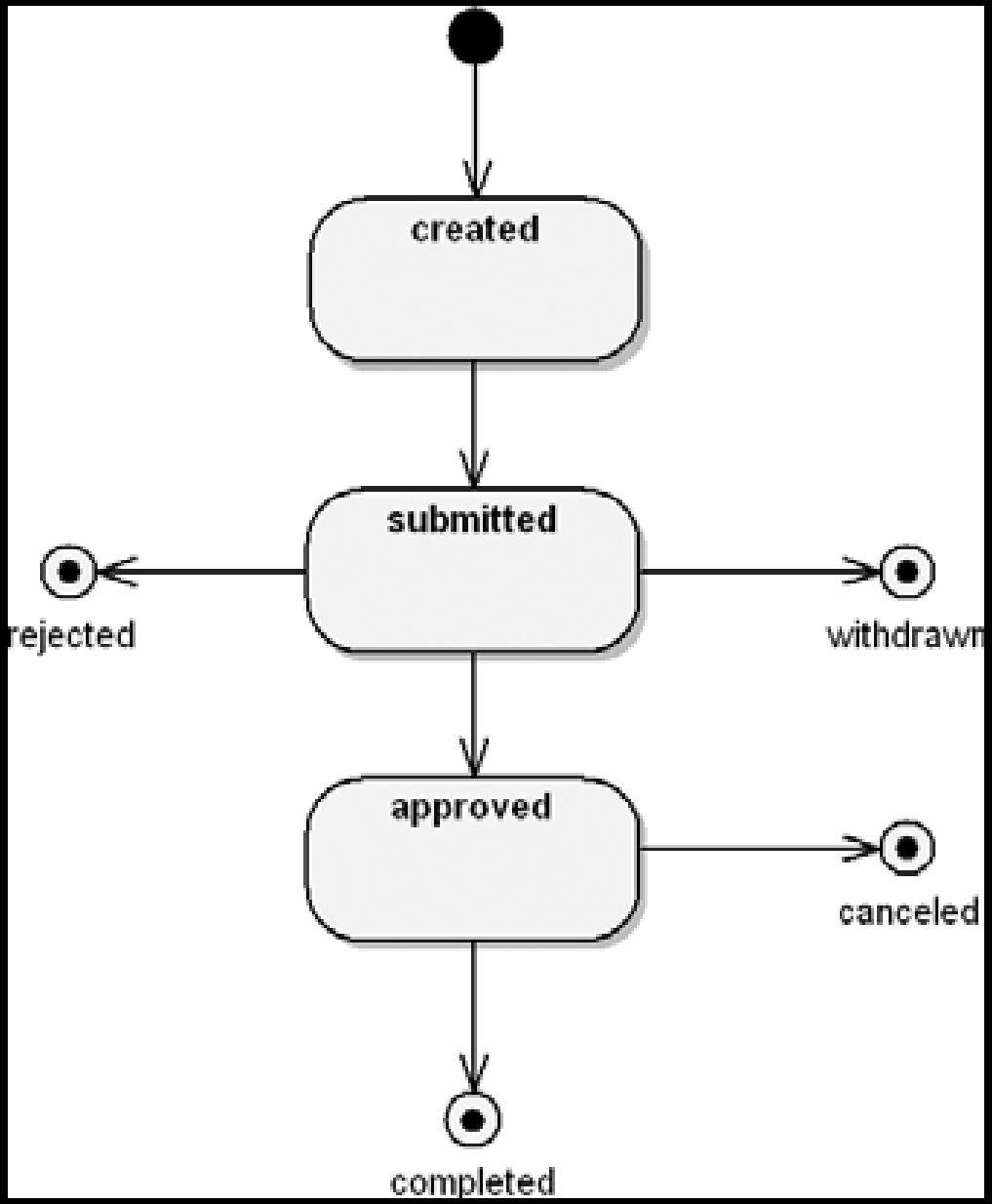
An Analysis Model Class Diagram Supporting the Primary Use Case

12.3 Construction

- Several important points in the class diagram
  - Vacation time requests are separate and distinct from vacation time grants.
    - A grant represents available time for the employee to draw on when requesting time off
  - A manager *is an* employee
    - An employee can have multiple managers
      - An approval of an employee's request for vacation time may come from multiple sources

12.3 Construction

The State Machine for the Request Class



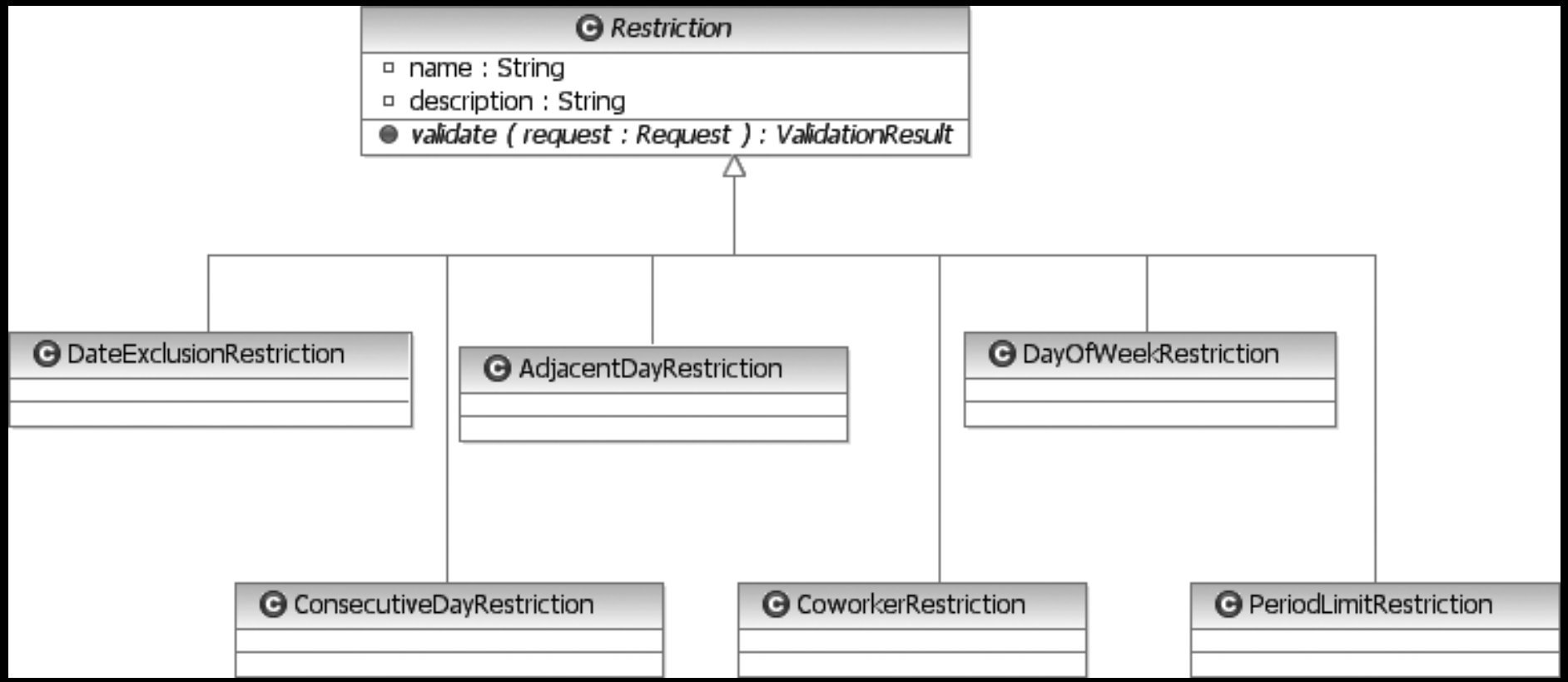
12.3 Construction

- The most difficult part of the analysis
  - Creating the elements related to ensuring that any requested vacation time passed all the rules and restrictions of the company and primary work location
    - Not stated in the use case specifications
    - But in the internal company documents
  - The requirements do state that there is to be a rules-based system managed by the HR department

12.3 Construction

- Summary of the major rule types.
  - An employee can't take more than *X* consecutive days of leave for *Y* type of grant.
  - Vacation time of type *X* cannot be taken when directly adjacent to a company or location-specific holiday.
  - Vacation time of type *X* is limited to *Y* hours per week or month.
  - Vacation time may not be granted when there are only *X* number of employees scheduled to work from the list *Y* of employees.
  - Vacation time may not be granted on these dates: *X*.
  - Vacation time of this type is limited to certain days of the week: {M, T, W, Th, F, Sat, Sun}.

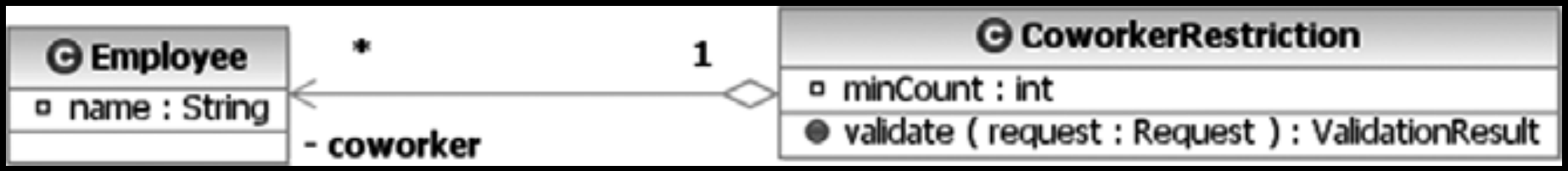
12.3 Construction



The Restriction Class Hierarchy

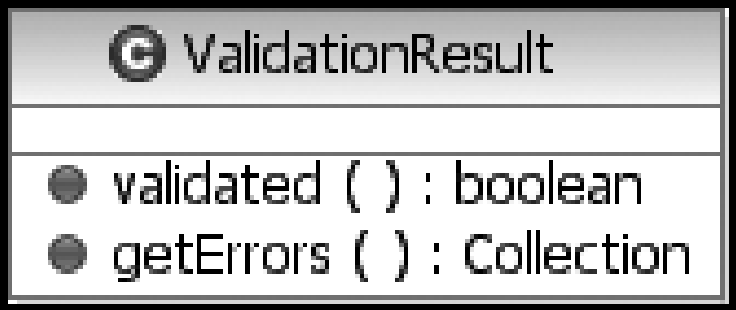
12.3 Construction

- Each specialization is required to implement the validate(),
  - Accepts a Request object as a parameter
  - From the Request object, the Restriction can navigate to most of the information it needs (Employee, Grant, Location)
  - Each specialization will manage a set of properties or relationships to objects that cannot be derived or navigated to



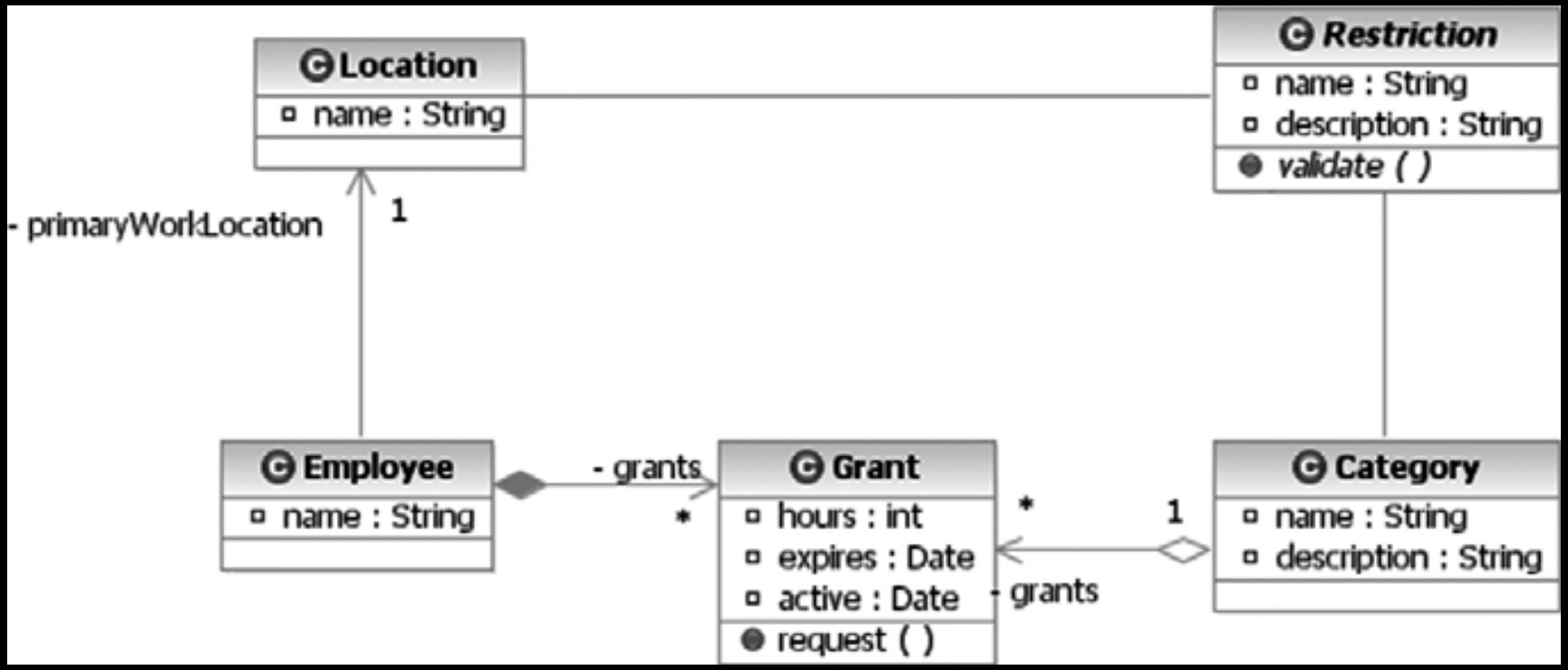
12.3 Construction

- validate() method initially returned just a simple Boolean
  - the user needs to be notified with an explanation



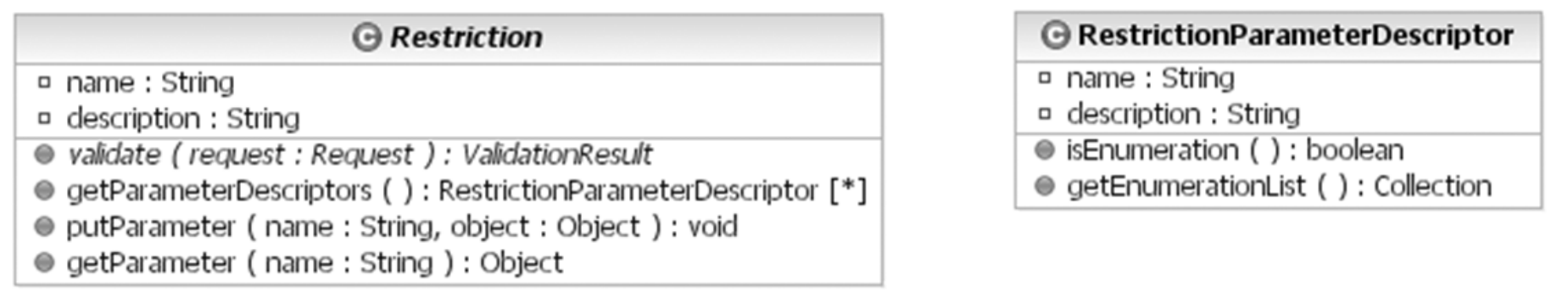
12.3 Construction

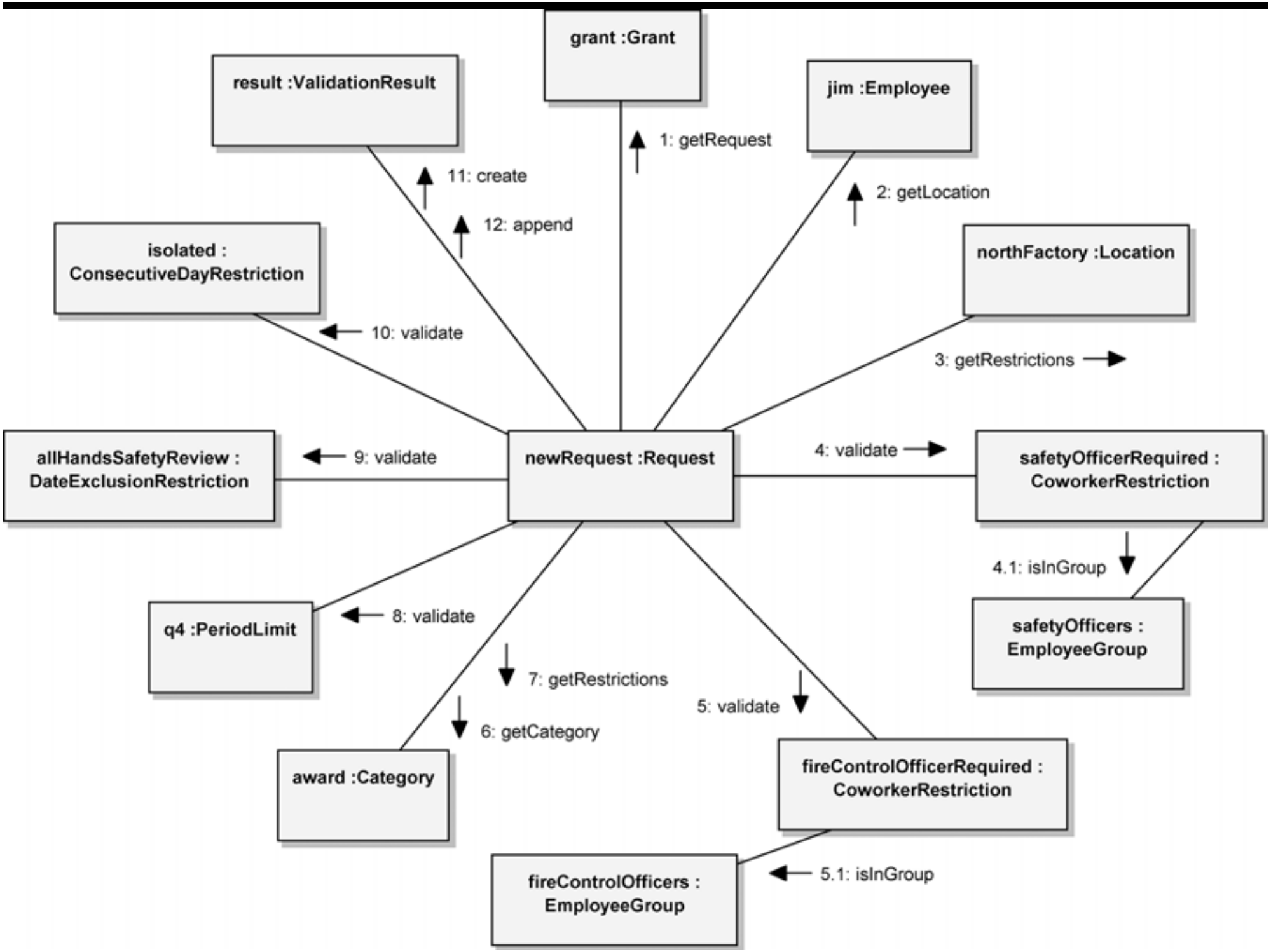
- Restrictions may not only be associated with a particular Grant or Employee object,
  - but also are broadly applied to a Location or Category of grants.



12.3 Construction

- Revisit the HR use cases
  - RestrictionParameterDescriptor is defined to encapsulate the values of parameters.
  - the abstract Restriction class should be able to accept and provide individual parameter values




grant :Grant
result :ValidationResult
jim :Employee
1: getRequest
11: create
12: append
2: getLocation
isolated :
ConsecutiveDayRestriction
northFactory :Location
10: validate
3: getRestrictions
allHandsSafetyReview :
DateExclusionRestriction
9: validate
newRequest :Request
4: validate
safetyOfficerRequired :
CoworkerRestriction
4.1: isInGroup
q4 :PeriodLimit
8: validate
safetyOfficers :
EmployeeGroup
7: getRestrictions
5: validate
6: getCategory
award :Category
fireControlOfficerRequired :
CoworkerRestriction
fireControlOfficers :
EmployeeGroup
5.1: isInGroup

12.3 Construction

- The project's architecture
  - *Client tier*
    - Any HTML 3.2 capable browser
  - *Presentation tier*
    - Java Server Pages (JSP), Servlets, and Java Server Faces (JSF)
  - *Business tier*
    - Enterprise Java Beans (EJB) 2.x (specifically, Container Managed Persistence [CMP] beans for entities, and session beans for controllers), and Service Data Objects (SDO)
  - *Security*
    - Central Authentication Service (CAS)