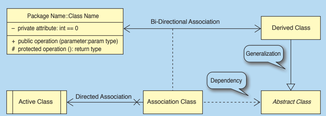
Architectural Analysis:

Purpose:
- To define the initial class structure required to support the system functionality

Artifacts:
- Classes defined by roles and responsibilities
- Class diagrams
- Initial class inheritance structure

Class Diagram:
It shows the static structure of the system in terms of the classes and their relations.



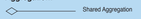
Association Adornments

Role and Label:

Multiplicity:

1	exactly one	mandatory
0..1	max. one	optional
*	many	optional
1..*	many	mandatory
n..m	min. n, max. m	

Aggregation:

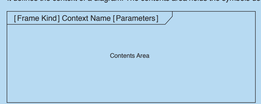


Qualifier:

Identifies instances of the target class

Ordering:

Diagram Frame:
It defines the context of a diagram. The contents area holds the symbols describing the context.



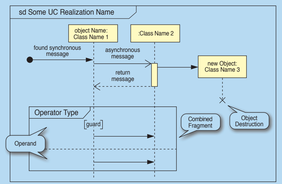
Frame Kind:
- cls (class)
- pkg (package)
- cmp (component)
- stm (state machine)
- act (activity)
- uc (use case)
- sd (interaction)

Use Case Analysis:

Purpose:
- To complete the class structure and to identify the mechanisms and dynamics required to realize the system functionality

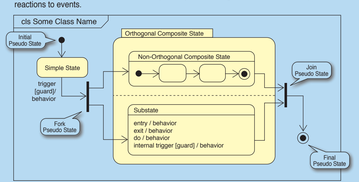
Artifacts:
- A use case realization for each use case
- Class diagrams and sequence diagrams showing how use cases are realized
- State machines for classes with state dependent behavior

Sequence Diagram:
It shows lifelines and their message exchange over time.



Operator Type:
- ref
- loop
- opt
- alt
- par
- strict
- critical
- assert
- neg

State Machine Diagram:
It shows the sequence of states that an object may have during its life and its state dependent reactions to events.

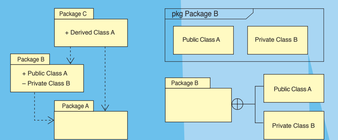


Logical Architecture:

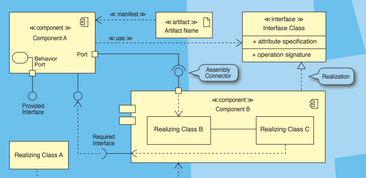
Purpose:
- To set the foundation for concurrent development
- To reduce dependencies between development units (packages and components)
- To expand the purely functional analysis model with IO related functionality

Artifacts:
- A logical architecture view showing packages, layers and subsystems in the form of package diagrams, component diagrams and component interface specifications

Package Diagram:
It shows packages, their contents as well as the overall structure and layering of the system.



Component Diagram:
It shows components, their ports, interfaces and realizing elements as well as how the components are wired. A component is a logical reusable SW unit, defined by provided and required interfaces. It is implemented by one or more manifesting artifacts.

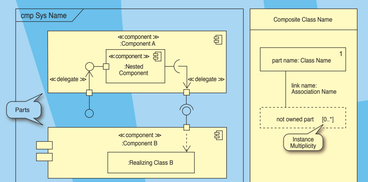


Physical Architecture:

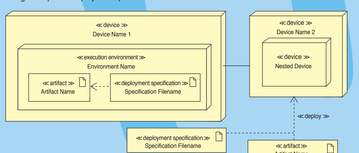
Purpose:
- To complete the design so that the system can be effectively realized
- To evolve the design and system architecture so that all requirements and constraints are satisfied

Artifacts:
- A physical architecture view showing threads, processes, artifacts and nodes in the form of composite structure, deployment, communication and timing diagrams

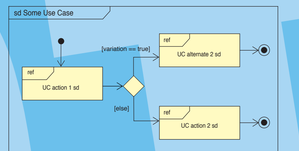
Composite Structure Diagram:
It shows the links and multiplicities of the composing parts of composite classes or components.



Deployment Diagram:
It shows the nodes, their relations and composition. It also shows the deployed artifacts along with optional deployment specifications.

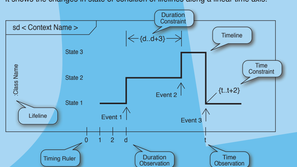


Interaction Overview Diagram:
It provides an overview of the flow of control between interaction diagrams.
It is like an activity diagram, whose nodes are interactions.

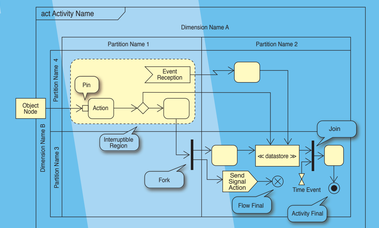


Communication Diagram:
It shows objects, their links and messages between them.

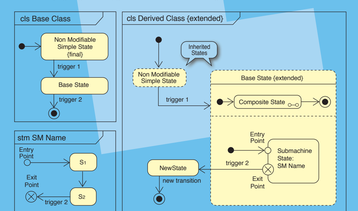
Timing Diagram:
It shows the changes in state or condition of lifelines along a linear time axis.



Activity Diagram:
It shows the sequence and conditions for coordinating actions. It also shows the related data and control flows, as well as event reception, time events and interrupts.



Advanced State Machine:
State machines and states can be extended by:
- adding new transitions, states and substates
- modifying target states, guards and actions of existing transitions



ANALYSIS DISCIPLINE

Purpose:
- To find a logical solution for the functional requirements

Artifacts:
- Complete and fully specified object oriented model, in the form of class diagrams, sequence diagrams, state machine diagrams and full class specifications

Process:

DESIGN DISCIPLINE

Purpose:
- To evolve a robust and flexible system architecture
- To satisfy all supplementary requirements (URPS and DIPI)

Artifacts:
- An architectural view highlighting the architecturally significant aspects
- A complete and fully specified object oriented model that includes package diagrams, component diagrams, composite structure diagrams, timing diagrams and deployment diagrams

Process:

Logical architecture		
Reduce class complexity and dependencies using principles of: ■ Open closed ■ Liskov substitution ■ Dependency inversion ■ Interface segregation	Define package architecture using principles of: ■ Reuse and common closure ■ Package dependency ■ Dependency direction ■ Package stability Define components and their interfaces.	Model system interfaces ■ Identify interface requirements from the use cases ■ Define a complete interface concept ■ Design and integrate interfaces with minimal changes to the functional model

Physical architecture		
Define the process view ■ Identify active classes ■ Model synchronization using timing diagrams ■ Model communication mechanisms	Define initialization of components and active classes ■ Draw composite structure diagrams	Define the physical system architecture ■ Identify physical artifacts and their contents ■ Identify nodes and assign artifacts

ACTL is your ultimate partner in the domain of object technology for state of the art system development, technology transfer and process improvement.

Our service portfolio encompasses:

- Outsourcing and headhunting services for highly qualified professionals
- Iterative project development
- Technology transfer through
  - Training
  - Project focused coaching and mentoring
- Process improvement
  - Assessments and Audits
  - Unified Process, Agile or SCRUM
  - Requirements Management
  - OOA/OOD
  - Project Management
  - Testing

Reduce your cost and raise your productivity.
Use our experience and knowledge to successfuly accomplish your products.
Join our high-profile customers list, together with IAI, Intel, Microsoft...

Reference Card

ACTL

Advanced CASE Technology & Language Systems

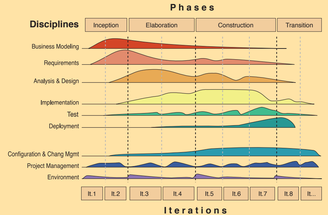
for
Requirements
Analysis & Design

ACTL
ACTL Systems Ltd.
Jaffa Street 217/5
P.O.B 8129
91081 Jerusalem
Israel

Tel: +972-2-537-6459
Fax: +972-2-537-0425
E-mail: info@actl.co.il
URL: www.actl.co.il

THE OO DEVELOPMENT PROCESS

The development process is:
- use case driven
- incremental & iterative
- architechture centric

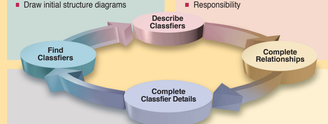


The Micro Process

It defines the core activities of object oriented analysis and design.

Find classifiers in problem domain
- Create an initial list of classifiers
- Select relevant classifiers
- Draw initial structure diagrams

Describe classifiers by
- Description
- Roles
- Responsibility



Complete classifier details
- Identify attributes
- Identify operations
- Identify state dependent behavior

Complete classifier relationships
- Describe associations
- Identify correct generalizations
- Identify dependencies and realizations

REQUIREMENTS DISCIPLINE

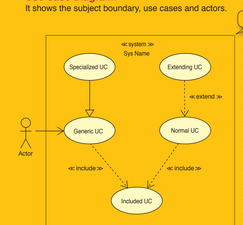
Purpose:
- To define system scope and boundaries
- To establish and maintain agreement with the customers and other stakeholders on:
  - what the system should do
  - conditions and constraints

Artifacts:
- Vision / Charter document
- Fully specified use case model (actors, use case diagrams and use case documents with optional activity and sequence diagrams)
- Supplementary requirement documents (URPS and DIPI)

Process:
- Write a vision / charter document
- Understand and model the problem domain
- Describe the system boundaries by identifying actors
- Describe system functionality by identifying and describing use cases
  - the textual specification can be enhanced with activity and sequence diagrams
- Describe the supplementary requirements

Use Case Diagram:

It shows the subject boundary, use cases and actors.

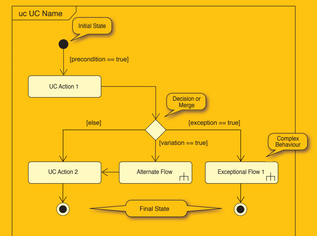


Use Case Description:

Each use case is textually described in the following manner:
- Title
- Goal
- Description
- Actors
- Pre-conditions
- Post-conditions
- Basic flow
- Exceptional flows
- Alternate flows
- Supplementary requirements

Activity Diagram:

It provides a graphical overview of the use case flow, its exceptional and alternate flows.



Sequence Diagram:

It provides a birds-eye view of the message exchange between the actors and the system.