UNIT- V
MANAGEMENT & PLANNING

Risk management

- The responsibility of the software development manager is to manage technical as well as nontechnical risk.

- Micro process of object oriented development is inherently unstable and requires active management.

- The macro process of object oriented development is designed to lead to closure by providing number of tangible products that management can study to ascertain the health of the project.

Task planning

- Task planning involves scheduling the deliverables of the macro process.
- Task planning at this level fails because of overly optimistic schedules.
- Task planning goes to specific part of the system.
- For e.g the design of a set of classes for interfacing to a relational database.

Staffing

- The object oriented project undertaken by an organization will require slightly more resources than for object oriented methods.
- For analysis resource requirements do not typically change much when employing object oriented methods.
- In the steady state the net of all the human resources required for object oriented development is less than that required for traditional approaches.

Release management

- **Integration:** industrial strength projects require the development of families of programs.
- The nature of interactive and incremental process of object oriented development means there should rarely be single “big bang “integration event
- For larger projects an organization may produce an internal release of system every few weeks.
- Release a running version to its customers for review every few months according to the needs of the project.
- Institutionalizing reuse: this means that opportunities for reuse must be actively sought out and rewarded

Reuse

- In object oriented languages classes serve as a primary linguistic vehicle for reuse.
- Classes may be sub classed to specialize or extend the base class.
- We can reuse patterns of classes, objects and designs in the form of idioms, mechanisms and framework.
- In the successful projects we have encountered reuse factors as high as 70% and as low as 0%.

Quality assurance and metrics

- Software quality: the fitness for use of the total software product.
- Object oriented technology doesn’t automatically lead to quality software.
- A simple adaptable architecture is central to any quality software.
- The defect discovery rate is thereby a measure of how quickly errors are discovered which we plot against time.
- a project under control have bell shaped curve
- Defect density is another quality measure.
- A healthy projects, defect reach a stable value after approximately after 10000 loc have been inspected and will remain unchanged no matter how long the code volume is.
- It is a more formal approaches to gather defect information.
- It is also useful to intitute project or company wide “bug hunts”.

Documentation and tools

- **Development legacy:** The development of a software system involves much more than writing of its raw source code.
- **Documentation contents:** it is an essential product of the development process.
- It should include: documentation of the high-level system architecture;
- Documentation of the key abstractions and mechanism in the architecture
- Documentation scenario that illustrate the as built behaviour of key aspects of the system
- Worst:
- worst possible documentation to create for an object oriented system is a stand alone description of the semantics of each method on a class by class basis.
- To generate a great deal of useless documentation architectural issues.

Tools

- The development team has a minimal tool set: an editor, a compiler, a linker, and a loader.
- It is important to choose tool that scale well.

There are 7 different kinds of tool

1. **Graphical based system:** supporting object oriented notation
2. **Browser:** to know about class structure and module architecture of a system
3. **Incremental compiler:** it is used for compiling single declarations and statements.
4. **Debugger:** that knows about class and object semantics
5. **Configuration management and version control tool:** this category or sub system is the best unit of configuration management.
6. **A class librarian:** it consists of predefined class libraries or commercially available class libraries.
7. **GUI builder:** interactively create dialogs and other windows than to create these artifacts from the bottom up in code.

Development team role

- The three roles to an object oriented project are:

  1.) Project architect

  2.) Subsystem lead

  3.) Application engineer

1. Project architect: is the visionary and is responsible for evolving and maintaining the systems architecture.

- For small to medium sized systems architectural design is typically the responsibility of one or two particularly insightful individuals.
- Project architect is not necessarily the most senior developer, but he is one of the best qualified to make strategic decision.

Development team role

2. **Subsystem Lead:** A subsystem lead is therefore the ultimate owner of a cluster of classes.

* they are usually faster and better programmers than the project architect.

* on the average the subsystem leads constitute about a third to a half of the development team

Development team role

3.) **Application engineer:** Application engineers are the less senior developers in a project and carry out one of two responsibilities.

- Certain application engineers are responsible for the implementation of a category or subsystem
- Application engineers are familiar with but necessarily experts in the notation and process of object oriented development.
- The break down of skills addresses the staffing problem faced by most software development organizations.
- They only have really handful of really good designers and many more less experienced ones.

In larger project some of the distinct development roles to carry out the work of the project

- **Project manager:** responsible for active management of the projects deliverables, tasks, resources, and schedules.
- **Analyst:** responsible for evolving and interpreting the end users requirement.
- **Reuse Engineer:** responsible for managing the projects repository of components and design
- **Quality assurance:** responsible for measuring the products of the development process.
- **Integration manager:** responsible for assembling compatible versions of released categories and subsystems.

MACRO LEVEL PROCESS

- Macro level process identifies view classes by analysing the use cases
- Here the interface object handles all communication with the actor but processes no business rule.
- It operates as a buffer between the user and rest of the business objects
- It lies inside the business layer and involves no interaction with actors.
- Example: computing employee overtime

  here overtime is an interface object

The view layer of macro process

- The view layer macro process consists of two steps:

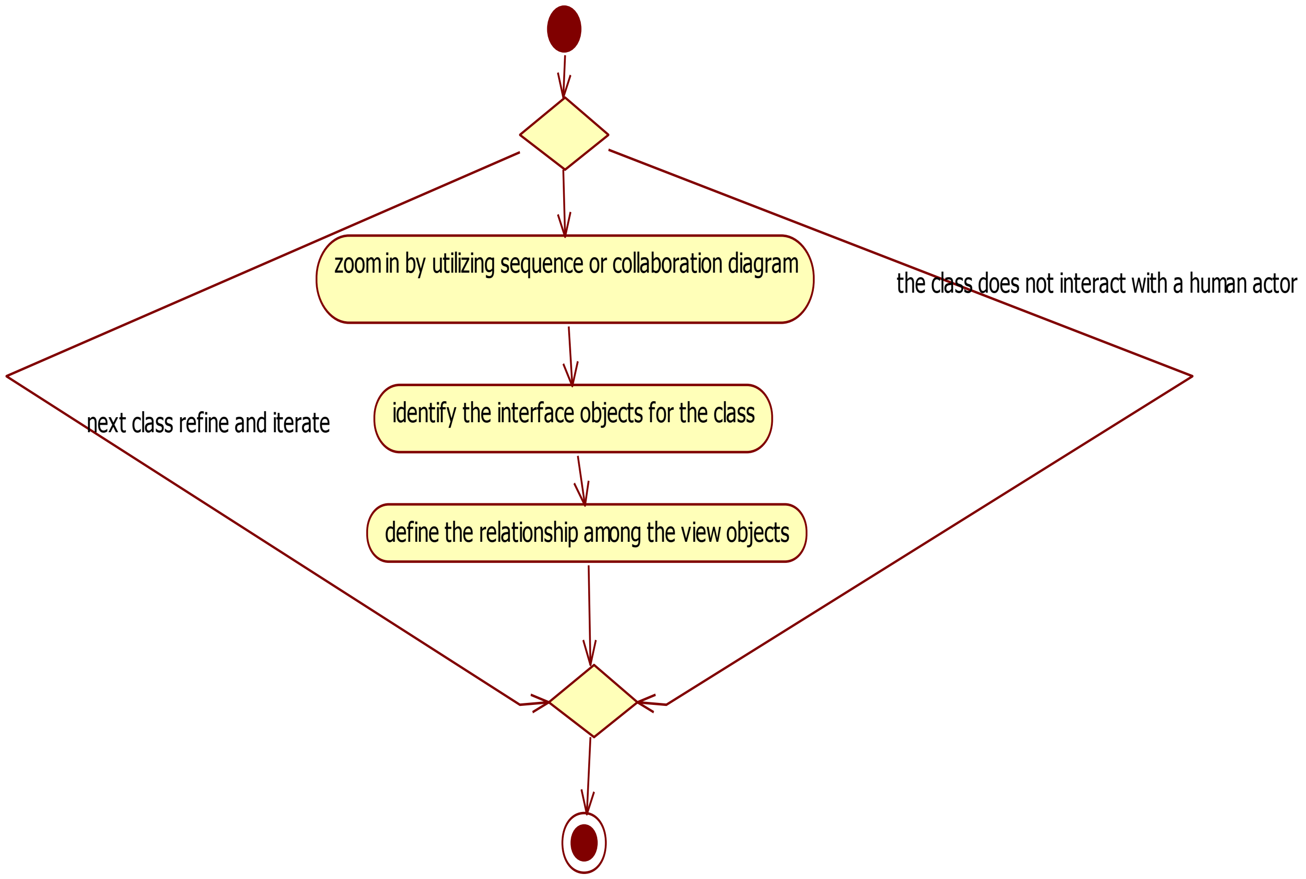
1. for every class identified find out if the class interacts with a human actor, if so

    * *identify the view (interface) objects for the class.*

    * *define relationship among the view (interface) objects.*

2. Iterate and Refine

Macro level design process


zoom in by utilizing sequence or collaboration diagram
the class does not interact with a human actor
next class refine and iterate
identify the interface objects for the class
define the relationship among the view objects

Micro Level process

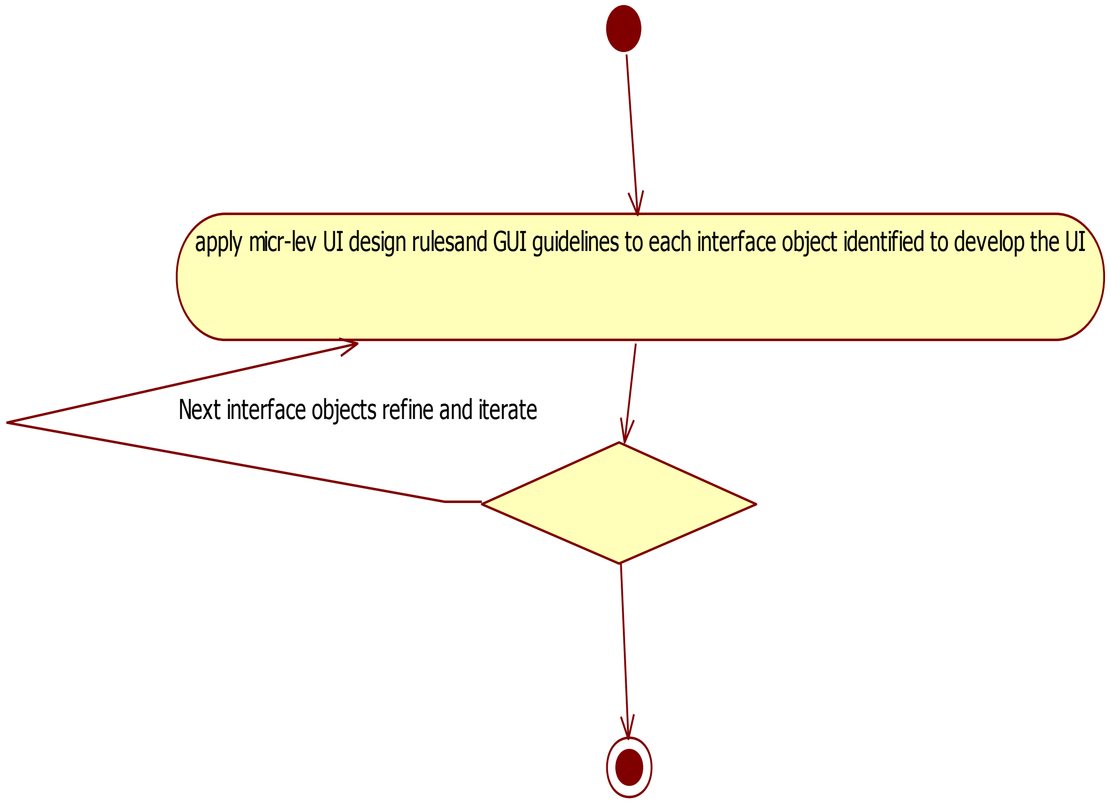
- Micro level process is made successful by designing the view layer objects.
- It must be user driven or user-centred.
- A user-entered interface *replicates the users view of doing things by providing the outcomes user expects for any action.*

Designing view consists of two process:

1. for every interface object identified in the macro UI design process apply micro –level UI design rules and corollaries to develop the UI

2. iterate and refine

Micro level design process


apply micr-lev UI design rulesand GUI guidelines to each interface object identified to develop the UI
Next interface objects refine and iterate

Purpose of View Layer interface

- User interface can apply one or more windows. Windows commonly used for the following purposes:
- Forms and data entry window-used to retrieve, display, and change in application.
- Dialog boxes – display status information or ask users to supply information.
- Application windows (main window)- it contains an entire application with which the user can interact

UI design rule

1. Making the interface simple
2. Making the interface transparent and natural
3. Allowing users to be in control of the software

Software quality assurance

To deliver a robust system we need high level of confidence.

- Each component will behave correctly
- Collective behaviour is correct
- No incorrect collective behaviour will be produced.

Quality assurance test

- Quality assurance is needed because computers are in famous for doing what you tell them to do, not necessarily what you want them to do.
- To close this gap the code must be free of errors.
- Debugging is the process of finding error and eliminating them to avoid unexpected results

Types of error encountered when you run your program

- **Language(syntax):** occurs due to incorrectly constructed code, they are easy to detect, no debugging tools is needed mostly. The system will report the existence of these error.
- **Run-time error :** these errors are detected as the program is running , when a statement attempts an operation that is impossible to carry out.
- **Logic error:** when a code does not perform the way you intended . The code might be syntactically valid and run without performing any invalid operation and yet produce incorrect result . Only by testing the code and analysing the results can you verify that the code performs as intended logic error can also produce run time error.

Release management

Integration:

- Industrial strength projects require the development of families of programs.
- Development is done by many smaller integration events each marking the creation of another prototype or architectural release
- Each such release is generally incremental in nature having evolved from an earlier stable release.
- For larger projects an organization may produce an internal release of the system every few weeks and then release a running version to its customers for review every months, according to the needs of the project.
- Release is possible whenever the major subsystems of projects are stable enough to work together well to provide some new level of functionality.

Release management

Configuration management &version control:

- Consider stream of releases from the perspective of an individual developer, who might be responsible for implementing a particular system.
- He or she must have a working version of that sub system.
- As the working version becomes stable it is released to an integration team for collecting a set of compatible sub systems

Testing strategies

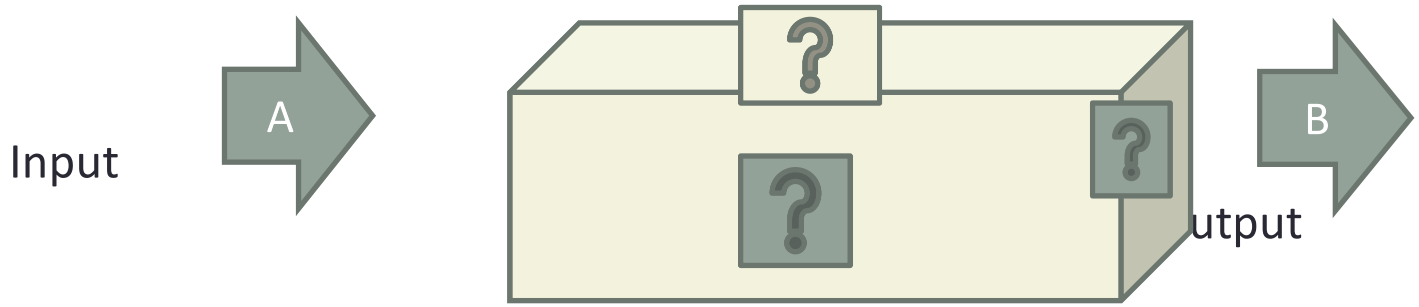
- Testing a system is controlled by many factors
- Such as the risks involved, limitations on resources, and deadlines.
- The most common testing strategies use combination of black box testing, white box testing, top-down testing, and bottom up testing.

Testing

- **Unit testing:** It involves testing individual classes and mechanisms, it is the responsibility of the application engineer who implement the structure.
- **Subsystem testing:** subsystem tests can be used as a regression test for each newly released version of the subsystem.
- **System testing:** It involves testing the system as a whole responsibility of quality-assurance team system tests are also typically used as regression tests.

Black box testing

- In black box the test item is treated as “black” since its logic is unknown
- You try various inputs and examine the resulting output; you can learn what the box does but nothing about how this conversion is implemented.



- Black box testing works very nicely in testing objects in an object-oriented environment.
- It can also be used in scenario based tests, the system ‘s inside may not be available for inspection .
- but the input and output are defined through use cases.

White box testing

- White box testing assumes that the specific logic is important and must be tested to guarantee proper functioning.
- The main use of white box test is it is error based testing.
- One form of white box testing is called path testing.
- Two types of path testing are i) statement testing coverage and ii) branch testing coverage.
- statement testing coverage : the main idea of statement testing coverage is to test every statement in the objects method by executing it at least once.
- Branch testing coverage : the main idea behind branch testing coverage is to perform enough tests to ensure that every branch alternatively has been executed at least once under some test.

Top-down testing

- The main logic or objet interactions and systems messages of the application need more testing
- It can detect the serious design flaws early in the implementation.
- It supports testing the user interface and event-driven systems.
- This approach is useful for scenario-based testing , top down testing is useful to test sub system and system integration.

Bottom –up testing

- Bottom up testing starts with the details of the system and proceeds to higher levels by a progressive aggregation until they collectively fit the requirements.
- This approach is more appropriate for testing the individual objects in a system.
- Here each object is tested and their interaction and the messages are passed among objects by utilizing top down approach.
- Here it starts with methods and classes that call or rely on no others.
- Bottom-up testing leads to integration testing which leads to system testing.

Test plan

- A **test plan** offers a road map for testing activities. It should state the test objectives and how to meet them
- The following steps are needed to create a test plan:

1. objectives of the test. Create the objectives and describe how to achieve them.

2. Development of the test case: develop test data, both input and expected output based on the domain of the data and the expected behaviours that must be tested.

3. Test analysis: this step involves the examination of the test output and the documentation of the test results.

if the bugs are detected, then this is reported and activity centres on debugging.

After debugging steps 1 through 3 must be repeated until no bugs can be detected.

All passed test should be repeated with the revised program called regression testing.

According to Tamara Thomas a test planner at Microsoft, a good test plan is one of the strongest tools you might have.

Who should do the testing ?

- For small application, the designer or the design team usually will develop the test plan and test cases.

- Many organizations have a separate team such as quality assurance group, that works closely with the design team and is responsible for such activities.

Beta testing & alpha testing

- **Alpha testing:** It is done by in-house testers, such as programmers, software engineers and internal users.
- **Beta testing:** It is a popular in-expensive and effective way to test software on a select group of actual users of the system.

Reference

- Object Oriented Analysis and Design with applications – Grady Booch (2nd edition)