Actualization

1. Actualization
   1. Programmers implement the new functionality
      1. According to change request
   2. The process of actualization varies
      1. Depends on the size of the change
2. Small changes
   1. Done directly in old
3. Larger changes
   1. Programmers implement the new classes separately from the old code
   2. The new code is plugged into the existing code
      1. Incorporation
   3. The change can propagate to other components of the system
      1. Ripple effect
4. Polymorphism
5. Adding new component
   1. Implement the new classes separately from the clients in the old code
      1. The new classes assume the responsibilities demanded by the change request
   2. The new classes are plugged as components into the appropriate place of the existing code (incorporation)
   3. Change propagation
6. Deletion of obsolete functionality
   1. Also causes change propagation
   2. All references to the deleted functionality must be deleted
      1. Secondary changes propagate to other classes
7. Understand impact set
   1. Impact analysis estimates which classes are impacted
   2. Change propagation modifies the code of impacted classes
      1. Change propagation is the moment of truth
      2. It confirms or refutes the predication of impact analysis
      3. The accuracy of impact analysis predictions is important for software managers
8. Precision
   1. Used in the information retrieval
   2. Precision = (true positive)/(true positive + false positive)
9. Recall
   1. Recall = (true positives)/(true positives + false negatives)
   2. The programmers estimated that the changes will impact only about a third of all classes that actually changed
   3. Missed the other two third
10. Underestimation
    1. Common in SE
       1. Consequence of invisibility
    2. Make planning difficult
    3. Common in other field also