Software Development CYCLE
in Astronomy

Eszter Pozna
ESO
Vienna Dynamics Workshop 8-11 October 2019
What is software engineering?

- Convert ideas to software
- Deliver on time
- Keep it running..(reliably, efficiently)
Its about...

\[ m_i \frac{d \mathbf{r}_i}{dt} = \sum_{j \neq i}^{N} \frac{G m_i m_j (\mathbf{r}_i - \mathbf{r}_j)}{||\mathbf{r}_i - \mathbf{r}_j||^3} \]
Project pitfalls...

How Projects Really Work (version 1.5)

1. How the customer explained it
2. How the project leader understood it
3. How the analyst designed it
4. How the programmer wrote it
5. What the beta testers received
6. How the consultant customized it
7. What the project was documented
8. What operators installed
9. How the customer was billed
10. How it was supported
11. What marketing advertised
12. What the customer really needed

Create your own cartoon at www.projectcartoon.com
THE SOFTWARE DEVELOPMENT CYCLE

1. PLANNING
2. ANALYSIS
3. DESIGN
4. IMPLEMENTATION
5. TESTING & INTEGRATION
6. MAINTENANCE

SDLC
Process for Small project

Big Bang Model

- This model is ideal for small projects like academic projects or practical projects. One or two developers can work together on this model.
- In this model, developers do not follow any specific process. Development begins with the necessary funds and efforts in the form of inputs.
- And the result may or may not be as per the customer's requirement, because in this model, even the customer requirements are not defined.
- [https://www.javatpoint.com/software-engineering-big-bang-model](https://www.javatpoint.com/software-engineering-big-bang-model)
Prototyping

- Planning
- Analysis
- Design
- Development
- Test
- Maintenance

- Requirements

- triaxalSchwarzschild
- 10K LOC
- Complex concepts
- Less formality

Eszter Pozna - Vienna Dynamics Workshop 8-11 October 2019
Modelling stellar dynamics

- parameter fitting
- Select the best model
Parallel computing is perfect for modeling, simulating.

How?

make better use of underlying parallel hardware

Cluster of computer
When?
Requirements

Planning

Analysis

Design

Development

Test

Maintenance

Requirements
Analysis

- Feasibility study
  - Details
  - Prototyping/Testing
  - Discuss GUI

- Check availability reusability of existing software

- Cost, Time
Document your idea in a way that is meaningful for software people (UML, Activity diagram)
Analysis

- Concept via UML activity diagrams
Milestones

- First prototype
- Parallel execution
- Best Fit Loop
- Data handling
- C++ wrappers
- New features: models C++
- triaxial Schwarzschild via C++
- GPU
- Optimisation
- AI
- 100% C++
- User req
- MAC
- GUI
- tests
Design

➢ Architecture
  • Packages
    – classes
  • Connections

➢ Data handling (memory and file handling)
➢ Flexibility/ Configuration etc.
➢ Logging and debugging
➢ User interface
Software Design

- Design patterns
- Design decisions (e.g. process or thread)
Free stuff

- Platform: Linux by Linus Torvalds
- GIT version control
- Language: C++ by Bjarne Stroustrup
- IDE
- Download support Libraries
- Installation method (Makefile, Maven, Gradle, Waf,..)
- Problem reporting system (Jira, Bugzilla)
- Setup Database
80 % of the time the software runs in 20% of the code (Pareto rule)

- Stability is given by the rest!!
  - Error handling, Special cases,
  - preparation, logging,
  - startup/shutdown actions

- Unit test

- Documentation
Software development practices

- DRY
- KISS
- robust
- modular
Software Challenges

- Interface Fortran/C++
- Parallelization
  - Race conditions
- Memory handling
- Flexibility via design pattern
- Stability
- Optimization for speed
Test and integration

- Test: phaseA (developer) → phaseB (professional testers) → phaseC (requester)
- Integration: Test platform → User platform
MAINTENANCE
Maintenance

- repeatedly updating software for various reasons
  - improvement: Involve changes in functionality
  - Adaptative: Changes in the environment are adapted to the requirements
  - Corrective: Activities for error correction
  - Preventive: Improvements to avoid future problems
MAINTENANCE COST

- 50% and 70% of all total costs

Diagram showing distribution of maintenance costs with percentages:
- Requirements: 3%
- Designing: 8%
- Implementation: 7%
- Testing: 15%
- Maintenance: 67%

Lehman's Laws

- "Continuing Change"
- "Increasing Complexity"
- "Declining Quality"