Introduction to Software Architecture

“In the beginning, everything was one big lump of machine code”

Topics:

- Software life cycle
- Modularity
- Clarity
- Composition/Separation
- Simplicity
- Robustness
- Representation
- Interface design
- Optimization
- Scalability
- Attitude

Software Life Cycle

- Requirement/Specification
- Design
- Implementation
- Testing
- Maintenance

Modularity

- Simple parts connected by clean interfaces.
- Encapsulation
- Compactness
- Orthogonality
- Single point of Truth (*SPOT* Rule)

Modularity

- Approaches for software design
  - Top down
  - Bottoms-Up
- Good practices
  - Few global variables
  - Smaller function size
  - Functions and data structure forming a unit hiding implementation details.
  - Fewer entry points per module

Clarity

- Transparency
- Discoverability
- Good practices
  - Readability of source code
  - Properly commented and indented
  - Documentation
  - Avoiding magic numbers
  - Avoiding magic flags

Composition

- Programs connected with other programs
- Representing data in text format
- Handing off tasks to Specialist Programs

Separation

- Separate interface from engines
- Multi tier applications
- Distributed Applications

Simplicity

- The notion of “intricate and beautiful complexities” is almost an oxymoron.
- Avoid over designing
- “Big” programs increase maintenance cost.

Robustness

- Program should perform under unexpected conditions.
- Robustness is a child of transparency and simplicity.
- Avoid special cases.

Representation

- Move complexity from code to data.
- Design data structures first.
- Let data control the program flow.

Rule of Least Surprise

- In interface design do least surprising thing.
- Keep intended audience in mind.
- User interface should be easily understandable.
- Delegate or emulate interface functions of familiar programs.

Optimization

- Programmer time is expensive.
- Use higher-level languages.
- Write programs to write programs.
- Make it run, then make it right, than make it fast.

Scalability

- Avoid designing rigid softwares.
- Leave room for data formats and code to grow.
- Make data layout self describing.

Attitude matters

- Never be rigid about your design.
- Be open to humiliation.
- Be curious.

Resources

- http://www.google.com
- The Art of Unix Programming – Eric S. Raymond
- Object Oriented Analysis and Design – Grady Booch
- The Elements of user experience – Jesse James Garrett

Thank You