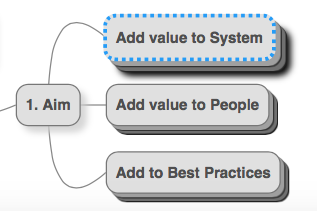
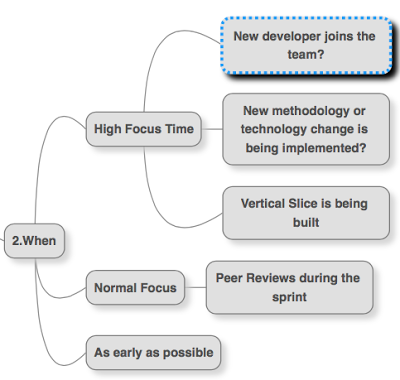
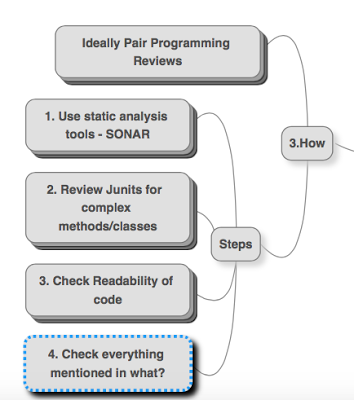
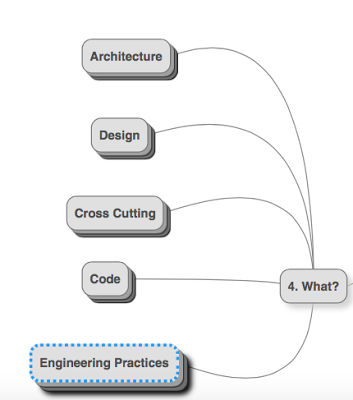
Code Review Interview Questions

[](http://1.bp.blogspot.com/-V4KIO66_rss/VVDswRx_hrI/AAAAAAAAAKM/nZSUOaxNbNk/s1600/Code%2BReview%2BInterview%2BQuestions.png)

* 1. What is the aim when you do code review? [](http://3.bp.blogspot.com/-ZoMolPgZ8Vo/VVDsuM1UHNI/AAAAAAAAAJ8/xor3PHn_-5c/s1600/Code%2BReview%2B1%2B-%2BAim%3F.png)
  1. Add value to System
     + Maintainability
     + Operations
     + Scalability
     + Performance
  2. Add value to People
     + Help them learn new things
  3. Add to Best Practices
     + Identify common mistakes/patterns
* 2. When do you do code review? [](http://4.bp.blogspot.com/-tnmt2t6VR6A/VVDsudpf7wI/AAAAAAAAAKc/H1iuZrV0Q94/s1600/Code%2BReview%2B2%2B-%2BWhen%3F.png)
  1. I do a highly focused code review when:
     + New developer joins the team?
     + New methodology or technology change is being implemented?
     + Vertical Slice is being built
  2. Normal Focus
     + Peer Reviews during the sprint
  3. Most important thing about code reviews is that they should be done as early and as often as possible
* 3. What are the good practices in doing Code Reviews?[](http://2.bp.blogspot.com/-j-NYUOIDNjU/VVDsurn-_aI/AAAAAAAAAKA/LhkLfrZOwYo/s1600/Code%2BReview%2B3%2B-%2BHow%3F.png)
  1. Ideally Pair Programming Reviews
     + Fixing comments directly discussing with developers
  2. Steps
     + 1. Use static analysis tools - SONAR
       - components & their sizes & interactions with other components
       - Hotspots
         * Large Classes
         * Complex methods
         * Large Components
         * Lot of dependencies
         * Uncovered Code
       - Static Analysis Tools are a very good start - but I would not just depend on static analysis tools for code review
     + 2. Review Junits for complex methods/classes
       - I think quality of Junit is a great guide to the quality of system
       - Makes all the dependencies very clear
     + 3. Check Readability of code
       - Most important of the 4 principles of Simple Design
     + 4. Check everything mentioned in the fourth category : what?
* 4. What are the things you look for in the Code Review?[](http://3.bp.blogspot.com/-_uWhiSCenWI/VVDsvsZXu9I/AAAAAAAAAKE/WHig9XLY_6w/s1600/Code%2BReview%2B4%2B-%2BWhat%3F.png)
  1. Architecture
     + Frameworks choice
     + Communication with other sytems
     + Testability
     + Components
       - Relative size
     + Reuse
       - Existing components
       - Creating new components
  2. Design
     + Interactions between classes
       - Coupling
       - Cohesion
     + Layer Responsibilities
       - Web Layer
         * Business logic creeping in?
       - Assemblers
       - Business Layer
       - Access Layer
     + OOPS principles
       - Domain Models
     + Unit Tests
       - How easy is it to unit test?
  3. Cross Cutting
     + Reuse
     + NFRs
       - Scalability
       - Performance
       - Security
       - Needs of operations team
         * Logging etc
  4. Code
     + 4 principles of simple design
     + Basics
       - Formatting
       - Size
       - Complexity
       - New language features
       - Documentation
         * Javadoc
     + Scalability
       - Bottlenecks
     + Performance
       - NO premature optimizations
       - Object creation in loops
       - Closing connections and other open stuff
       - Session usage
       - Caching?
       - Database related?
     + Operations
       - Proper Exception Handling
       - Transaction Management
     + Security
     + Unit tests
       - Readable
     + Langauge Specific
       - Object class methods
         * equals
         * hashcode
       - String
       - Generics
       - Enums
       - Creating and destroying objects
  5. I also try get a hang of the Engineering Practices followed.
     + How often is code committed?
     + Release practices
     + Broken Builds
     + Deployment Practices
     + Continuous Integration