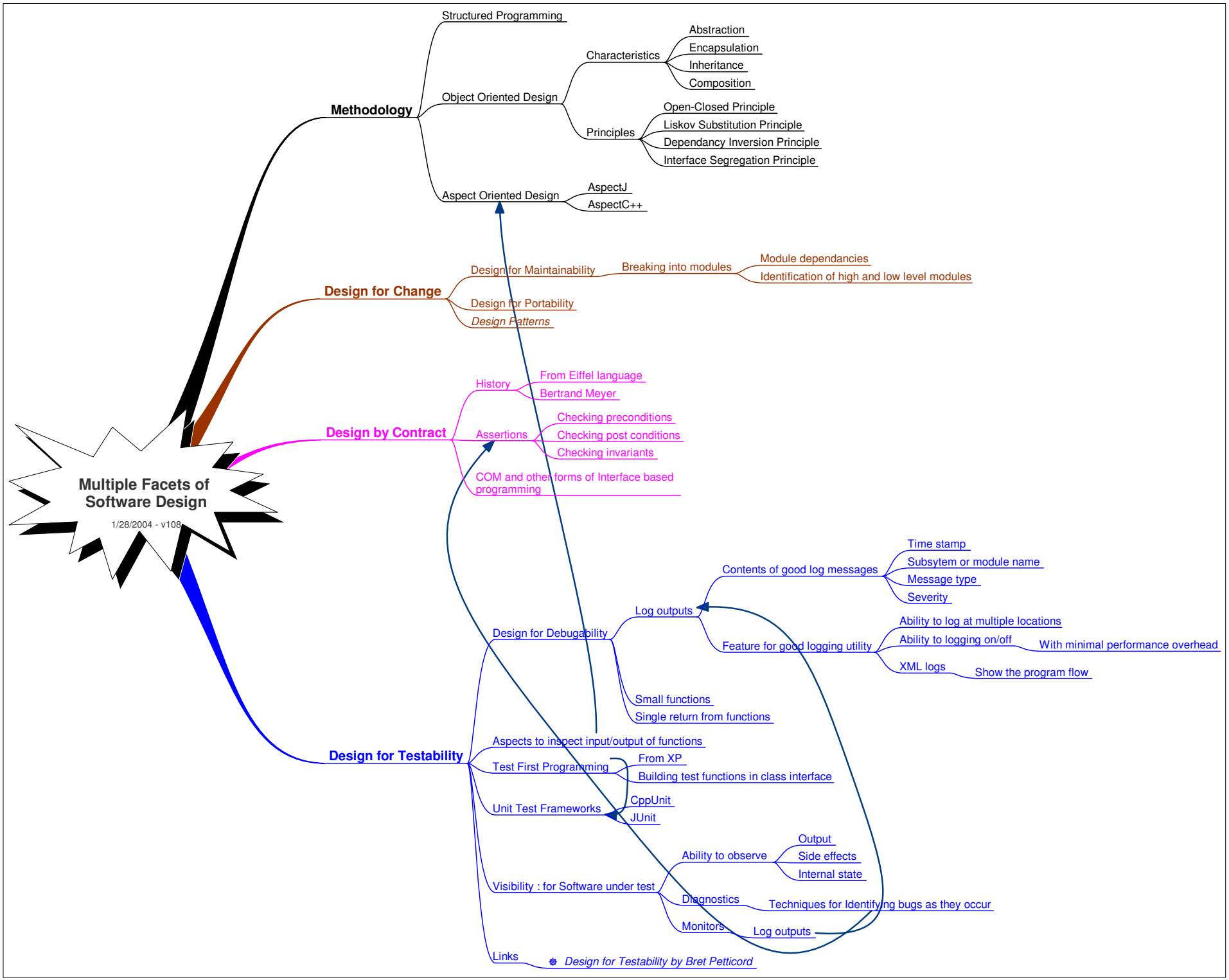
Multiple Facets of Software Design
1/28/2004 - v108
Methodology
Structured Programming
Object Oriented Design
Characteristics
Abstraction
Encapsulation
Inheritance
Composition
Principles
Open-Closed Principle
Liskov Substitution Principle
Dependancy Inversion Principle
Interface Segregation Principle
Aspect Oriented Design
AspectJ
AspectC++
Design for Change
Design for Maintainability
Breaking into modules
Module dependancies
Identification of high and low level modules
Design for Portability
Design Patterns
Design by Contract
History
From Eiffel language
Bertrand Meyer
Assertions
Checking preconditions
Checking post conditions
Checking invariants
COM and other forms of Interface based programming
Design for Testability
Design for Debugability
Log outputs
Contents of good log messages
Time stamp
Subsytem or module name
Message type
Severity
Feature for good logging utility
Ability to log at multiple locations
Ability to logging on/off
With minimal performance overhead
XML logs
Show the program flow
Small functions
Single return from functions
Aspects to inspect input/output of functions
Test First Programming
From XP
Building test functions in class interface
Unit Test Frameworks
CppUnit
JUnit
Visibility : for Software under test
Ability to observe
Output
Side effects
Internal state
Diagnostics
Techniques for Identifying bugs as they occur
Monitors
Log outputs
Links
Design for Testability by Bret Petticord