

Log Concept, Code Review and Common Defect



Section 1

LOGGING – LOG4NET

- If you've been writing code for any reasonable amount of time, then it's virtually impossible that you haven't handled logging in any way, since it's one of the most essential parts of modern, "real life" app development.
- If you're a .NET developer, then you've probably used some of the many famous logging frameworks available for use at this platform.
- Log4net is a logging framework for the .NET platform. It's definitely not the only one, but it's one of the most popular frameworks out there.
- A logging framework is a tool that can dramatically reduce the burden of dealing with logs.

LOG levels




How to work with log4net


NuGet: ConsoleApp4 Program.cs


Browse Installed Updates

log4net ☐ Include prerelease


Package source: nuget.org

 **log4net** by The Apache Software Foundation, **50.6M** downloads v2.0.8


 The Apache log4net library is a tool to help the programmer output log statements to a variety of output targets.

 **Microsoft.ApplicationInsights.Log4NetAppender** by Microsoft, **4.04M** downloads v2.14.0


Application Insights Log4Net Appender is a customer appender allowing you to send Log4Net log messages to Application Insights. Application Insights will collect your logs from multiple sources and provide rich powerful s...

 **Log4Net.Async** by Chris Haines, **1.16M** downloads v2.0.4


Asynchronous Log4Net appenders and forwarder

 **Abp.Castle.Log4Net** by Abp.Castle.Log4Net, **1.06M** downloads v5.7.0

Abp.Castle.Log4Net

 **Microsoft.Extensions.Logging.Log4Net.AspNetCore** by Huor Swords, **1.84M** downloads v3.1.0

Allows to configure Log4net as Microsoft Extensions Logging handler on any ASP.NET Core application.

 **log4net**

Installed: 2.0.8 Uninstall

Version: 2.0.8 Update

Options

Description

log4net is a tool to help the programmer output log statements to a variety of output targets. In case of problems with an application, it is helpful to enable logging so that the problem can be located. With log4net it is possible to enable logging at runtime without modifying the application binary. The log4net package is designed so that log statements can remain in shipped code without incurring a high performance cost. It follows that the speed of logging (or rather not logging) is crucial.

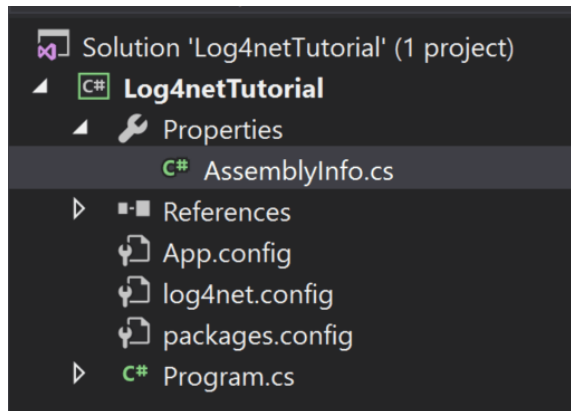
File App.config

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
    <section name="log4net" type="log4net.Config.Log4NetConfigurationSectionHandler,Log4net"/>
  </configSections>

  <log4net>
    <appender name="TestAppender"
      type="log4net.Appender.RollingFileAppender" >
      <file value="D:\log\MyTestAppender.log" />
      <encoding value="utf-8" />
      <appendToFile value="true" />
      <rollingStyle value="Date" />
      <!--<rollingStyle value="Size" />
      <maxSizeRollBackups value="5" />
      <maximumFileSize value="5MB" />
      <staticLogFileName value="true" />-->
      <layout type="log4net.Layout.PatternLayout">
        <conversionPattern value="%date %level [%thread] %type.%method - %message%n" />
      </layout>
    </appender>
  </log4net>
  <root>
    <level value="All" />
    <!-- If the following line is not included the log file
    will not be created even if log4net is configured with this file. -->
    <appender-ref ref="TestAppender" />
  </root>
</log4net>
<startup>
  <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.6.1" />
</startup>
</configuration>
```

Tell log4net to Load Your Config

- You can find it under the Properties section in your project:



- In AssemblyInfo.cs file

```
// Let log4net know that it can look for configuration in the default  
application config file  
[assembly: log4net.Config.XmlConfigurator(Watch = true)]
```

- Log Something

```
using System;

class Program
{
    private static readonly ILog log =
    LogManager.GetLogger(System.Reflection.MethodBase
    .GetCurrentMethod().DeclaringType);
    static void Main(string[] args)
    {
        log.Info("Hello logging world!");
        Console.WriteLine("Hit enter");
        Console.ReadLine();
    }
}
```


Section 2

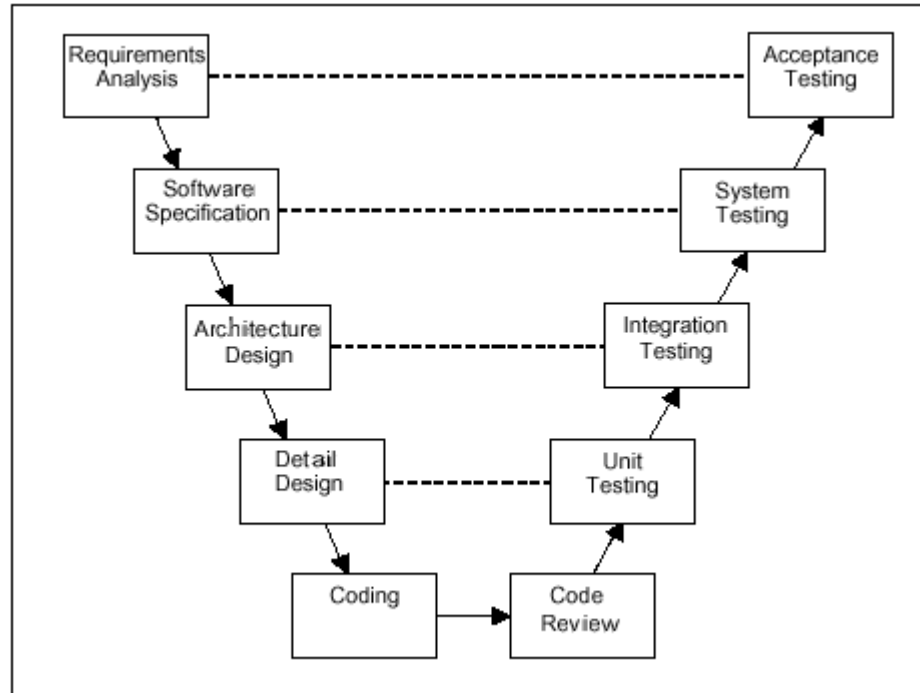
CODING PROCESS

Introduction

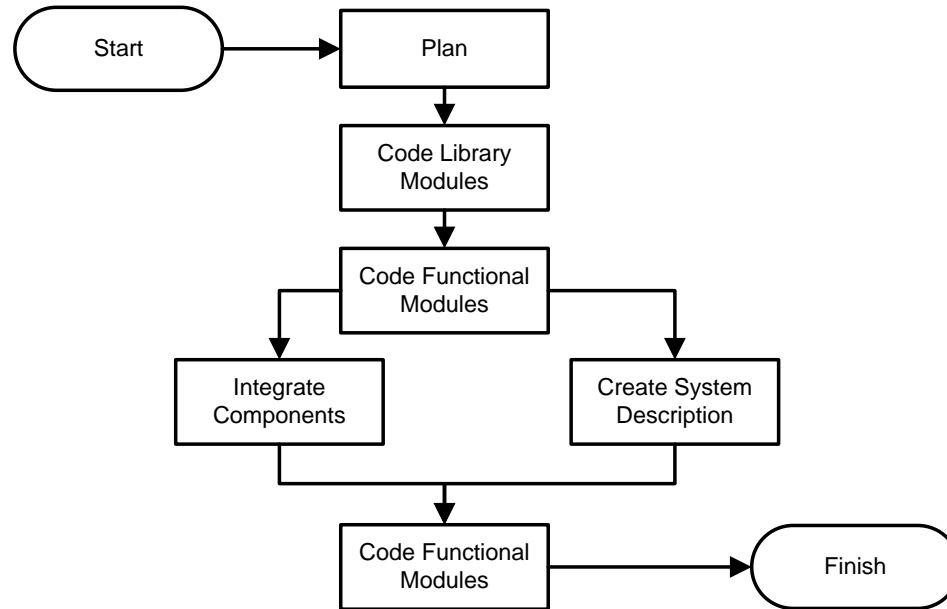


- Coding Process
- Code Review Process

Where the Coding is?



Coding Workflow



- **Purpose:** To plan and prepare for coding
- **Purpose:** To plan and prepare for coding
 - ✓ Study design documents
 - ✓ Define and prepare resources and infrastructure for coding, unit test and integration, if necessary.
 - ✓ Create coding plan including targets, scope, required deliverables and acceptance criteria, task and schedule, responsibilities
 - ✓ Review and obtain agreement on coding plan
 - ✓ Develop/customize coding convention
 - ✓ Review & conduct training on coding convention
 - ✓ Verify tools support for coding (if any)

Coding Library Modules

- **Purpose:** To build, construct and/or develop library modules
- **Steps:**
 - ✓ Create detail design for library modules
 - ✓ Code library modules
 - ✓ Review code of library modules
 - ✓ Fix defects of library modules
 - ✓ Summarize related documents

Coding Functional Modules

- **Purpose:** To build, construct and/or develop functional modules
- **Steps:**
 - ✓ Create detail design for modules and program units, if required in design documents
 - ✓ Code modules and program units
 - ✓ Review code
 - ✓ Fix defects for modules and program units
 - ✓ Summarize and submit result to Team Lead

Integrate Software Modules

- **Purpose:** assemble the software package from the software modules, ensure that the software package, as integrated and functions properly
- **Steps:**
 - ✓ Create integration plan (if needed)
 - ✓ Integrate modules
 - ✓ Evaluate integration results, conduct cause analysis, raise change request (if needed)
 - ✓ Review and approve results of integration

Create System Description

- **Purpose:** To develop System Description /User Manual documents that support in software operation
- **Steps:**
 - ✓ Make overview on system
 - ✓ Describe sub-systems and main functions including system structure scheme, flow charts, system interfaces, data flows
 - ✓ Describe system requirements including support data, memory capability, CPU and I/O requirements, storage capability, data for internal and external interfaces
 - ✓ Describe software structure including library of source codes (for system, sub-systems, objects) library of executive and supporting programs
 - ✓ Develop User Manual
 - ✓ Review and approve System Description/User Manual

Deliver & Summarize

- **Purpose:** To deliver software package
- **Steps:**
 - ✓ Review, do final inspection and summarize software products including documents
 - ✓ Deliver to test team
 - ✓ Create coding summary report
 - ✓ Maintain documents, records

Section 3

CODE REVIEW

- A thorough technical & logical line-by-line review of a code module (program, subroutine, method, ..)
- In a code review, the team would examine a sample of code and fixes any defects in it
 - ✓ Codes is inspected to identify possible improvements and ensure that business requirements are met.
 - ✓ For example, it could be made more readable or its performance could be improved

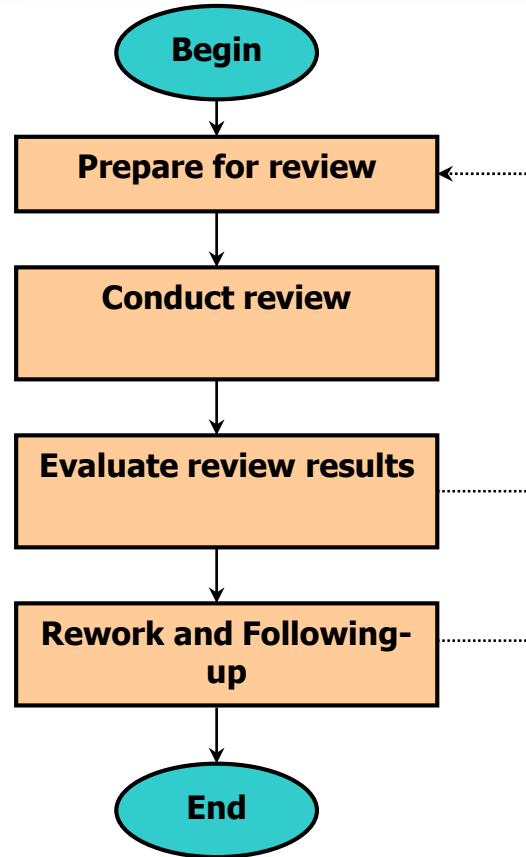
Review Benefits

- A different perspective
- Assess and accelerate progress
- Pride/reward: more pride,
- Project/module familiarity
- Fewer bugs and less rework,
- Better team communication
- Team cohesiveness

Review Candidates

- A portion of the software that only one person has the expertise to maintain
- Code that implements a highly abstract or tricky algorithm
- An object, library or API that is particularly difficult to work with
- Code written by someone who is inexperienced or has not written that kind of code before, or written in an unfamiliar language
- Code which employs a new programming technique
- An area of the code that will be especially catastrophic if there are defects

Review Workflow



Reviewing Inputs

- Statement of objectives of the review
- Source codes to be reviewed
 - ✓ Must be completed, tested by developer
 - ✓ Must be completed, tested by developer
- Other affected project components: documentation, test cases, a project schedule, or requirements changes, etc.
- Other affected project components: documentation, test cases, a project schedule, or requirements changes, etc.

Prepare for review 1 - 2

- Preparing agenda and facilities for the review
- Contacting with PM/QA to identify list of reviewers
- Sending agenda along with all inputs to reviewers
- If a meeting is necessary, gathering all comments /questions of reviewers and prepare answers/solutions for each one



Prepare for review 2 - 2

- Preparing notes:
 - ✓ Reviews are conducted as needed, usually based on the rate of code output.
 - ✓ The frequency of individual participation in a peer review depends primarily on the size of the programming team.
 - ✓ A team of 3 developers might include all three in every review.
 - ✓ Larger teams might be able to rotate participation based on experience, skill level, subject matter familiarity, ...
 - ✓ The review should include the programmer, two reviewers, a recorder, and a leader.
 - ✓ Other considerations for the size of the review team might be the scope of the project, workload, or training needs.

Conduct Review 1 - 2

- Performing online review. Some questions reviewers might bring up:
 - ✓ How does this module actually satisfy the stated requirement?
 - ✓ How does the output affect the previously base-lined interface documentation?
 - ✓ Wouldn't a case statement work here instead of a nested if-then-else structure?
 - ✓ Etc.
- Logging and sending to conductor all defects found and comments/questions

Conduct Review 2 - 2

- Conducting the review meeting (if any).
 - ✓ The leader opens with a short discussion of the goal of the meeting and lays out any ground rules.
 - ✓ Prioritising all defects and comments/ questions during the meeting
 - ✓ The developer goes through and explains his/her code.
 - ✓ The reviewers raise, discuss on the issues, comments, questions
 - ✓ The developer addresses those issues and explains the logic, problems, and choices that resulted in this code
- Logging all new defects or unsolved questions found during the review meeting

Evaluate review & Follow up

- Evaluating review results
 - ✓ Preparing and issuing Review report to all attendees
 - ✓ Making approval or reject basing upon the review's results
- Rework and Follow-up
 - ✓ Fix code-review defects
 - ✓ Monitoring to make sure all defects are closed as planned
 - ✓ If necessary, preparing for the new review again.

Review Outputs

- Review Report: reviewer list, defect list, statistic and analysis...
- Filled-up checklists
- Minute of meeting (if any)
- Approval or Reject of approver

Self- Code Review 1 - 3

- What: developer to do self-code review while he/she do the coding, it is to make sure that:
 - ✓ Requirement logics are implemented correctly
 - ✓ No coding conventions or common defects existed
 - ✓ General programming practices are applied
- How:
 - ✓ Use code review tools
 - ✓ Use team-defined code review checklist

Self- Code Review 2 - 3

- **Code Review Tools**
- http://en.wikipedia.org/wiki/List_of_tools_for_static_code_analysis
- .NET
 - ✓ FxCop <http://msdn.microsoft.com/en-us/library/bb429476%28v=vs.80%29.aspx>

Self- Code Review 3 - 3

- Code Review checklist
 - ✓ This is a team-defined coding checklist.
 - ✓ Project developers are required to self review their codes following defined checklist items, filled the code review checklist as reviewing results
 - ✓ Main checklist items
 - General coding conventions
 - Code module, class commenting
 - Source code details: modulation, code structure, loop, naming conventions, comments, etc.

Section 4

COMMON DEFECT

Every software developer deals with defects. The really tough defects are the ones not detected by the compiler.

Nasty defects manifest themselves only when executed at runtime.

This topic is list of the top common defects



- Issue with giving a fixed value in codes, for example:

```
dgrView.PageSize = 10  
strErr = "Error message here";
```

The problem occurs when you should change these values multiple times!!!

- Preventive Action: define constants in the common constant module or in a configure files

The Dangling else Problem

- Issue with below codes?

```
if (x == 0)
    if (y == 0) error();
else {
    z = x + y;
    f (&z);
}
```

Confused on the else using!!!

- Cause: else is always associated with the closest unmatched if.
- Preventive: use appropriated braces ({})

- Issue: the developer got Null-Pointer-Exception run-time error, while he/she did not detect that when compiling the codes
`strReturn = objDoc.SelectNodes(strName);`
- Cause: the developer does not check null or think about null object before accessing object's value.
- Preventive: Should check null before accessing object or pointer before using its member

```
If (objDoc != NULL)  
    strReturn = objDoc.SelectNodes(strName);
```

- Code redundant issues:
 - ✓ Create new Object while we can reuse the object in previous command:
`BeanXXX bean = new BeanXXX();`
`bean = objectYYY.getBeanXXX();`
 - ✓ Variables are declared in based class but it is not used
 - ✓ Un-used methods/functions are existing in the application
 - ✓ Break a complex method/function to more simple methods / functions with only one or two lines of code, and could not be re-use
- Preventive actions:
 - ✓ Should verify that the current design is possible and is the best by coding sample
 - ✓ Re-check unnecessary code to remove in coding review
 - ✓ Supervise and assign person to review code carefully before coding
 - ✓ Supervise strictly changing source code from team daily

- ❑ **Issue** with variables or create objects in Loop?

```
for (int i=0; i<dt.Rows.Count-1; i++){  
    String strName;  
    strName = dt.Rows[i]["Name"].ToString();  
    //do something here  
}
```

Impact to the application performance!!!

- ❑ **Cause**: memory is allocated repeatedly.

- ❑ **Preventive**:

- ✓ Variables should be declared before the loop statement or inside for() statement
- ✓ Determine objects before loop statement

- ❑ **Issue** *Use string concatenated in loop:*

```
string stNumber = "";  
for(i=0; i<100; i++){  
    stNumber = stNumber + i;  
}
```

- ❑ **Cause:** Don't understand class String in C#.

- ❑ **Preventive:** Use StringBuilder instead

Check String empty

- ❑ **Issue** : Using the operator “==”

```
if(stringVariable == "") {  
    // do smt here  
}
```

- ❑ **Preventive** :

// Correct answer:

```
if (string.IsNullOrEmpty(stringVariable)) {  
    //do something here  
}
```

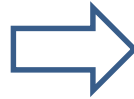
- ❑ **Issue:** Memory saving errors adversely affect system performance
- ❑ **Preventive:**
 - ✓ File operations: file read operations must be restricted to a minimum
 - ✓ Be economical when creating new objects

Errors by using Try/Catch

- Do not use try ... catch nested

// Wrong

```
Try {  
    Try {  
        ...  
    } catch() {  
        ...  
    }  
} catch () {  
}
```



// Right

```
Try {  
    ...  
} catch () {  
    ...  
} catch() {  
}
```

Thank you

