INFORMATIKOS FAKULTETAS

Programų sistemų testavimas (T120B162)

Atliko:

IFF-8/5 gr. studentai

Paulius Balčiūnas

Kasparas Jasiūnas

Priėmė:

Eduardas Bareiša

# Lab. 3. Static code analysis

## First part:

### Review checklist

1. Structure
   1. Is there any unnecessary code or can be simplified?
   2. Are classes have their own files?
   3. Does the code follow C# coding culture?
2. Documentation
   1. Methods have their own description.
   2. Classes have their own description.
3. Variables
   1. Are all variables properly defined with meaningful, consistent, and clear names?
   2. Are all variables following the C# coding culture?
4. Loops and Branches
   1. Does every case statement have a default?
   2. Does the code in the loop avoid manipulating the index variable or using it upon exit from the loop?
   3. Does the code avoid nested loops?
5. Defensive Programming
   1. Are indexes, pointers, and subscripts tested against array, record, or file bounds?
   2. Is every memory allocation de-allocated?
6. Readability
   1. Is the code self-explanatory?
7. Reliability
   1. Does the code have exception handling?
8. Extensibility
   1. Is it easy to add or change the code without major changes?
9. Usability
   1. Is it easy to use implemented class?

### Review several classes

Selected classes for review:

1. ServerEngine.cs
2. PlayerControl.cs
3. NetworkManager.cs

Issues that we found:

*“Didn’t use pattern matching”:*

**ServerEngine.cs –**

foreach (IObject newObject in newObjects)

{

newObject.GUID = Guid.NewGuid().ToString();

newObject.Init();

instantiadedObjects.Add(newObject.GUID, newObject);

if(newObject is NetworkObject)

{

NetworkManager.AddNewObjectToGroup(newObject.AreaId, (NetworkObject)newObject);

}

}

*“‘new‘ expression can be simplified“:*

**ServerEngine.cs –**

private readonly object ObjectProccessLock = new object();

public List<IObject> waitingObjects = new List<IObject>();

private Dictionary<string, IObject> instantiadedObjects = new Dictionary<string, IObject>();

**PlayerControl.cs –**

private List<ICommand> commands;

private Player player;

**NetworkManager.cs –**

private readonly object ServerRequestProccessLock = new object();

private readonly object ClientRequestProccessLock = new object();

private readonly object ProccessNetworkObjectLock = new object();

private const string ClientRequestHandlerMethod = "ClientRequestHandler";

private IHubContext<ChatHub> GameHub;

private List<NetworkObject> networkObjects = new List<NetworkObject>();

private List<NetworkRequest> allClientsRequestQueue = new List<NetworkRequest>();

private Dictionary<string, List<NetworkRequest>> clientGroupRequestQueue = new Dictionary<string, List<NetworkRequest>>();

private Dictionary<string, List<NetworkRequest>> singleClientRequestQueue = new Dictionary<string, List<NetworkRequest>>();

private List<NetworkRequest> clientsRequestQueue = new List<NetworkRequest>();

*“Fields should be readonly“:*

**NetworkManager.cs –**

private List<NetworkObject> networkObjects = new List<NetworkObject>();

private List<NetworkRequest> allClientsRequestQueue = new List<NetworkRequest>();

private Dictionary<string, List<NetworkRequest>> clientGroupRequestQueue = new Dictionary<string, List<NetworkRequest>>();

private Dictionary<string, List<NetworkRequest>> singleClientRequestQueue = new Dictionary<string, List<NetworkRequest>>();

private List<NetworkRequest> clientsRequestQueue = new List<NetworkRequest>();

**ServerEngine.cs –**

private long updateDelay = 300;

**PlayerControl.cs –**

private List<ICommand> commands;

private Player player;

### Class review

**ServerEngine.cs**

* **Structure:**

Classes should be separated intro different files.

Code can be simplified

* **Documentation:**

Next to nothing documented.

* **Variables**
* **Loops and Branches,**
* **Defensive Programming,**

There are no null checks in the code.

Memory isn’t deallocated.

Correct state upon termination isn’t used

* **Readability**

Most of the code could be written a lot better. Poor quality. Raw.

* **Reliability**
* **Extensibility**
* **Usability**

**NetworkManager.cs**

* **Structure**

Classes should be separated intro different files.

Contains unused code

Code can be simplified

* **Documentation:**

Next to nothing documented.

* **Variables,**
* **Loops and Branches,**
* **Defensive Programming,**

Memory isn’t deallocated

Correct state upon termination isn’t used

* **Readability**

Some of the code could be written a lot better.

* **Reliability**
* **Extensibility**

A lot of complex code would be difficult to add or change any parts.

* **Usability**

**PlayerControl.cs**

* **Structure**

Code can be simplified

* **Documentation:**

Nothing is documented.

* **Variables,**
* **Loops and Branches,**
* **Defensive Programming,**

Memory isn’t deallocated

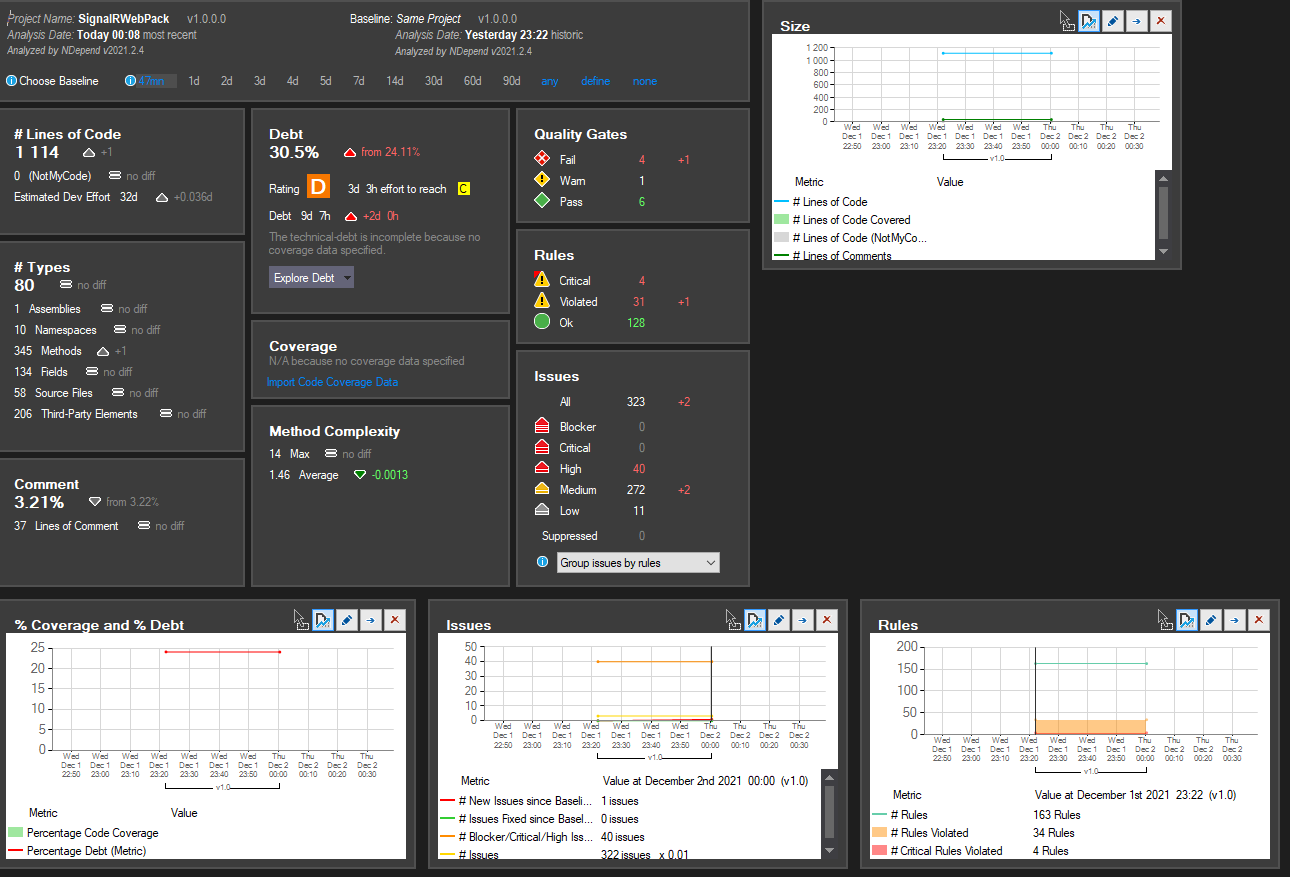
Correct state upon termination isn’t used

* **Readability**
* **Reliability**
* **Extensibility**
* **Usability**

## Static code analysis tool

Paveikslėlis, kuriame yra žinutė

Automatiškai sugeneruotas aprašymasAnalysis tool – NDepend



## Static analysis rule implementation

**Rule description** – Methods that have a ‘Get’ prefix in their name, should return some sort of a value

**Implemented rule code:**

*// <Name>Methods prefixed with 'Get' should return something</Name>*

warnif count > 0

**from** m **in** Application.Methods **where**

   m.SimpleNameLike("^Get") &&

   m.ReturnType.FullName == "System.Void"

**select** new {

   m,

   m.ReturnType,

   m.ReturnType.FullName,

   Severity = Severity.Medium

}

Paveikslėlis, kuriame yra žinutė, monitorius, vidinis, ekrano nuotrauka

Automatiškai sugeneruotas aprašymas

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Source file | Class | Method | Line | Rule | Comment |
| NetworkManager.cs | NetworkManager | GetGroupRequestDataCount() | 107 | Structure | Methods that start their name with ‘Get’, should return a value. |
| NetworkManager.cs | NetworkManager | GetSingleClientRequestDataCount() | 114 |
| NetworkManager.cs | NetworkManager | GetTotalRequestDataCount() | 121 |
| World.cs | World | GetPlayers() | 48 |
| World.cs | World | GetNpcs() | 58 |
| World.cs | World | GetItems() | 68 |
| World.cs | World | GetObstacles() | 78 |