Technische Universiteit
Eindhoven
University of Technology

Project 2IM24 - 2008-2009 (groep 2)

Project Etaris

Software Kwaliteit Garantie Plan

Haan de, S
Ng, HS
Penterman, AGJ
Roulaux, R
Vliembergen van, RWL
Hilkens, RGM

19 november 2008
v 0.1.7

Gr2 Dev

Where innovation starts

Inhoud

1 Documentrichtlijnen 2
1.1 Inleiding 2
1.2 Microsoft Word-documenten 2
1.3 Microsoft Excel-documenten 3
1.4 Algemeen 3
2 Overige richtlijnen 4
2.1 Naamgeving 4
2.2 E-mail 4
3 Richtlijnen programmeercode 5
3.1 Definities 5
3.2 Opbouw C#-oplossing 5
3.3 Naamgeving entiteiten 7
3.4 Algemene opbouw 8
3.5 Overig beleid 10
4 Etaris C#-oplossing 11
4.1 Pakketten 11
4.2 Kenmerken C#-oplossingen 11
5 Documentinspectie 12
5.1 Standaard procedure inspectie 12
5.2 Bijwerken niet-codebestanden 12
5.3 Bijwerken codebestanden 12
Bijlagen 14
Bijlage A, C#-entiteiten 14

1 Documentrichtlijnen

1.1 Inleiding

Om consistentie in documenten te waarborgen worden er voor ieder soort document richtlijnen opgesteld. In principe moet ieder document aan deze richtlijnen voldoen. Echter, met toestemming kan hiervan afgeweken worden. In algemeenheid kan gesteld worden dat een document wanneer mogelijk en niet strijdig met enig beleid, moet voldoen aan de hier gebruikte stijl.

1.2 Microsoft Word-documenten

1.2.1 Sjabloon

Tekstdocumenten worden altijd met Microsoft Word gemaakt in het 97-2003 bestandsformaat (.doc). Hiervoor is een sjabloon beschikbaar gesteld. Het sjabloon bevat o.a. deze kenmerken:

- Kleuren volgens het TU/e kleurenschema.
- Speciale kop- en voettekst met velden (zie hieronder)
- Gelijkend op de Microsoft Office 2007 stijl met diverse aanpassingen aangaande:
  - Lettergroottes
  - Kleuren
  - Alineaspatiëring

1.2.2 Stijlen

Voor veel terugkomende opmaak moet te allen tijde een stijl gedefinieerd worden. Vaak hebben visueel gelijkzijnde stijlen toch een aparte naam. Dit moet voorkomen worden. Het doel is om het aantal stijlen dat gebruikt wordt zo klein mogelijk te houden.

Alinea's en titels moeten aan voorwaarden voldoen. Voor normale alinea's wordt de stijl 'Normal' gebruikt. Wanneer het niet wenselijk is dat er witruimte na iedere alinea is, kan de stijl 'No Spacing' gebruikt worden. Voor titels worden de standaard stijlen 'Heading 1' tot en met 'Heading 3' gebruikt. Waarbij een titel op het hoogste niveau (een hoofdstuk) de stijl 'Heading 1' krijgt.

Kruisreferenties mogen opgenomen worden, een dergelijke verwijzing heeft altijd de stijl 'Fig Reference'. Onderschriften van figuren, tabellen en formules hebben altijd de 'Caption' stijl.

1.2.3 Standaardvelden en automatische nummering

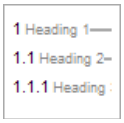

Fig. 1.1 Opsomming

Het sjabloon bevat velden in de kop- en voetteksten. Deze velden verwijzen naar de bedrijfsnaam, titel, wijzigingsdatum (formaat 'd MMM yyyy'), versie en auteur. Alleen de velden versie en auteur dienen aangepast te worden (zie § 5.2). Hercorrigeer brede '–' tekens in de koptekst naar een standaard minteken.

Automatische hoofdstuknummering is verplicht voor alle managementdocumenten (zie Software Project Management Plan). Dit kan door bijvoorbeeld op de eerste 'Heading 1' stijl een automatische opsomming in te stellen (zie Fig. 1.1). De titel van een inhoudsopgave heeft een 'Heading 1' stijl zonder nummering.

Een veld in Microsoft Word kan op diverse manieren bijgewerkt worden. Een geselecteerd veld kan bijgewerkt worden door F9 of via een andere Microsoft Word-functie, maar mag nooit eigenhandig door slechts tekst of een ander veld vervangen worden (inhoudsopgaventabellen mogen wel eigenhandig gemaakt worden door middel van velden, zie § 1.2.4).

1.2.4 Pagina indeling

Een Microsoft Word-document start met een titelpagina (zie § 1.4.1), gevolgd door een inhoudsopgaventabel en inleiding. Voor niet managementdocumenten is een inleiding (nog) niet verplicht. Hoofdstukken in managementdocumenten starten altijd op een nieuwe pagina. Voor

andere documenten is dit een aanbeveling. Inhoudsopgaven en inleidingen worden beschouwd als hoofdstukken.

De inhoudsopgaventabel moet altijd de tabelstijl hebben zoals weergegeven in dit document. De inhoudsopgaventitel komt zelf niet in de tabel voor. Wanneer met veel korte paragraven met 'Heading 3' titels gewerkt wordt (zoals deze), dan hoeven die niet opgenomen te worden in de hoofdstuknummering of inhoudsopgaventabel.

Alle bijlagen tezamen worden altijd geplaatst onder een titel 'Bijlagen' (een 'Heading 1' stijl). Individueel worden bijlagen genummerd met titels als (Bijlage A, …| Bijlage B, …| …). Titels met betrekking tot bijlagen worden nooit genummerd met cijfers.

1.2.5 Tabellen

Voor tabellen gelden geen specifieke richtlijnen. Echter, in de meeste gevallen is onderstaande opmaak hanteerbaar en moet indien mogelijk toegepast worden.

Kolom 1	Kolom 2	Kolom 3	Kolom 4
Cel 1			
Cel 2			

Tabel 1.1 Voorbeeld tabel

1.3 Microsoft Excel-documenten

1.3.1 Sjabloon

Eventuele spreadsheets worden altijd met Microsoft Excel gemaakt in het 97-2003 bestandsformaat (.xls). Hiervoor is een sjabloon beschikbaar gesteld. Het sjabloon bevat alleen kop- en voetteksten.

1.4 Algemeen

1.4.1 Titelpagina's

Een gedrukt document moet altijd een titelpagina hebben. Dit geldt ook voor Microsoft Word-documenten. Alleen de gegeven titelpagina's in PNG-formaat worden gebruikt, geen andere. Wanneer andere titelpagina's nodig zijn (i.v.m. een andere titel) dan kunnen deze worden gemaakt. Het PNG-bestand beslaat als afbeelding een geheel A4!

In de titelpagina voor Microsoft Office-documenten moet een tekstbox worden opgenomen met de wijzigingsdatum en versie. Het formaat van de datum is 'd MMMM yyyy'. Deze tekstboxen moeten wat betreft plaatsing en stijl voldoen aan de tekstbox op de titelpagina van dit document.

In Microsoft Word kan een titelpagina als afbeelding worden geplaatst. De grootte hiervan moet 100% zijn. De positie is absoluut t.o.v. de pagina en wordt geplaatst in de linkerbovenhoek.

1.4.2 Schrijfstijl, termen en afkortingen

De tekst moet toegespitst zijn op de doelgroep. Een hoofdstuk moet begrijpbaar zijn voor de doelgroep wanneer het in normale volgorde wordt doorgelezen. Het taalgebruik is formeel. Informeel taalgebruik is verboden.

In het algemeen mogen geen standaard afkortingen gebruikt worden. Wanneer het om specifieke afkortingen gaat, dienen deze voorgaande aan de afkortingen voluit geschreven te worden.

Een verklarende woordenlijst is niet verplicht. Een dergelijke lijst, indien aanwezig, moet altijd achteraan een document opgenomen worden en is alfabetisch geordend. Ook wanneer een dergelijke lijst aanwezig is, moeten termen en afkortingen toelicht worden wanneer zij voor het eerst gebruikt worden.

2 Overige richtlijnen

2.1 Naamgeving

Naamgeving van personen geschiedt altijd met <achternaam>, <voorletters> <tussenvoegsel(s)>. Hierbij worden de voorletters in hoofdletters geschreven, zonder punten ertussen. Voor naamgeving van personen in een alinea kan eventueel gebruik gemaakt worden van <voornaam> <tussenvoegsel(s)> <achternaam>. Dit laatste is informeler en verdient niet de aanbeveling.

Documenttitels (bijvoorbeeld in de koptekst) dienen overeen te komen met de subtitel van de titelpagina. Titels van managementdocumenten worden in het Nederlands geschreven. Andere titels worden indien van toepassing voluit in het Engels geschreven.

Voor eventuele subtitels gelden geen naamconventies met betrekking tot de Nederlandse of Engelse taal.

2.2 E-mail

Het onderwerp van een e-mail moet altijd starten met '2IM24 Minorproject, <titel>', waarbij <titel> het daadwerkelijk onderwerp van de e-mail is. Echter, bij beantwoorden en doorsturen mogen vóór het onderwerp zoals hier gedefinieerd de termen 'Re' en 'Fw' gebruikt worden.

E-mails welke betrekking hebben op alle teamleden of op het project in het algemeen dienen altijd als 'Carbon Copy' gestuurd te worden naar alle teamleden.

3 Richtlijnen programmeercode

3.1 Definities

Dit hoofdstuk beschrijft nauwkeurige richtlijnen omtrent de kwaliteit van de programmeercode.

Etaris moet geprogrammeerd worden in Microsoft Visual C# 2005/2008, kortweg C#. Dit dient gedaan te worden in de ontwikkelomgeving Microsoft Visual C# 2008 Express Edition (kortweg Visual Studio). In de projecteigenschappen moet compatibiliteit met .Net Framework 3.0 worden ingesteld. De code moet voldoen aan de C# specificatie versie 2.0. Gebruik van versie 3.0 is alleen toegestaan wanneer dit document dat expliciet eist. De C# specificatie is bindend en heeft voorrang wanneer er tegenstrijdigheid is met dit document. Deze specificatie is te vinden op: http://msdn.microsoft.com/en-us/library/kx37x362(VS.80).aspx.

Een 'solution' (oplossing) in Visual Studio is het hoofd projectbestand waarmee iets in Visual Studio geopend wordt. Een 'solution' kan diverse C#-projecten bevatten. Een C#-project is een eenheid binnen de oplossing en kan de code bevatten voor een uitvoerbaar programma of een bibliotheek: een assembly (doorgaans een gecompileerd .exe of .dll bestand).

Standaardbestanden refereren hieronder naar bestanden die de ontwikkelomgeving automatisch genereert bij een nieuw C#-project (exclusief *Program.cs*). Voor deze standaardbestanden en de automatisch gegenereerde bestanden eindigend op *.Designer.cs* gelden de richtlijnen in dit hoofdstuk niet. Wanneer er wel richtlijnen voor gelden, staat dit expliciet omschreven.

3.2 Opbouw C#-oplossing

Dit document stelt eisen aan iedere oplossing. Er is altijd een oplossing (SLN-bestand). De hoofdoplossing heet Etaris. Merk op dat in het geval van prototypes een aparte oplossing kan worden gemaakt, waarvan de naam in de regel wel begint met Etaris. De hoofdoplossing bevat de diverse C#-deelprojecten.

3.2.1 AssemblyInfo.cs

Het standaardbestand *AssemblyInfo.cs* bevat informatie over iedere assembly waaronder een beschrijving en versienummer. Deze informatie is gedeeltelijk zichtbaar in de bestandeigenschappen.

De volgende regels moeten veranderd worden in het standaard *AssemblyInfo.cs* bestand:

```
[assembly: AssemblyTitle("<TITEL, meestal ETARIS>")]
[assembly: AssemblyDescription("<KORTE BESCHRIJVING>")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("Gr2 Dev")]
[assembly: AssemblyProduct("Etaris")]
[assembly: AssemblyCopyright("© Gr2 Dev 2008")]
```

De attributen *AssemblyVersion* en *AssemblyFileVersion* moeten worden in ingesteld zoals beschreven staat in het Software Configuratie Management Plan.

3.2.2 Projecteigenschappen

In de standaard projecteigenschappen moet het volgende veranderd worden:

1. Target Framework: .NET Framework 3.0. De niet benodigde referenties in 'References' kunnen verwijderd worden (dit beleid kan later gewijzigd worden).
2. Default namespace: Etaris.

3.2.3 Program.cs

In geval van een uitvoerbaar programma moet de hoofd opstartroutine in *Program.cs* staan, eveneens een standaardbestand. Dit is altijd de functie: `static void Main()`.

3.2.4 Basisopbouw .cs bestand

Een C#-codebestand (*.cs*) dient altijd aan een aantal voorwaarden te voldoen.

Hieronder een voorbeeld:

```
/*** Name:       Program.cs
 *** Created:    9/23/2008, by: Hilkens, RGM
 *** Modified:   9/23/2008, by: Hilkens, RGM
 *** Source:     algemene opstartroutines.
 *** Copyright: (c) Gr2 Dev 2008
 */
<Naamruimte import>
using System;

namespace Etaris
{
    internal static class Program
    {
        <Globale root variabelen>
        internal static gui.frmMain frmMain;

        /// <summary>
        /// Opstartroutine.
        /// </summary>
        [STAThread]
        static void Main()
        {
        }
    }
}
```

Toelichting koptekst:

1. Deze koptekst is altijd verplicht.
2. Naamgeving van personen geschiedt altijd zoals eerder in dit document aangeven.
3. Name: bestandsnaam.
4. Created: datum van creatie en betreffende auteur.
5. Modified: datum van wijziging en betreffende auteur.
6. Source: korte beschrijving van het bestand. Doorgaans gelijk aan de beschrijving van de hoofdentiteit in het betreffende bestand.
7. Copyright: verplicht, wordt niet gewijzigd.
8. Datums worden altijd geschreven in het hier gegeven formaat: m/d/yyyy.

Bij voorkeur wordt iedere blokentiteit (zie Bijlage A, C#-entiteiten) in een apart bestand geschreven. Naar eigen inzicht kunnen meerdere entiteiten in een bestand worden geschreven. Dit kan nuttig zijn met een klasse en een set constanten (`enum`) welke bij elkaar horen. De naam van een codebestand is altijd de naam van de belangrijkste blokentiteit in dat bestand, gevolgd door *.cs*.

3.3 Naamgeving entiteiten

3.3.1 Algemene naamgeving

De naamgeving van een entiteit geschiedt altijd in het Engels. Een dergelijke naam is een korte passende Engelse naam voor de entiteit (zie Bijlage A, C#-entiteiten). Mocht een naam uit meerdere woorden bestaan, dan worden deze aan elkaar vast geschreven zonder scheidingssymbolen. Ieder woord begint met een hoofdletter. Aan onderstaande regels moet altijd voldaan worden, mits expliciet in dit document of door de kwaliteitmanager anders is aangeven. Zie onderstaande tabel voor de algemene naamgeving van entiteiten.

Entiteit	Beschrijving	Naamgeving	Voorbeeld
`namespace`	Een naamruimte. Nodig voor logische ordening.	Begint altijd met `Etaris`. Naamruimtes worden in overleg toegekend. Naamruimte voor venster en controllers is altijd `Etaris.Gui`.	`Etaris.Gui`
`class`	Een klasse.		
`interface`	Een interface.	Begint altijd met `I`.	`IList` `IPlayField`
`structure`	Een structuur		
`enum`	Enumeratie van constanten.	Bij voorkeur is de naam een meervoud. Een variabele van het type `enum`, voldoet aan de eisen van een variabele, maar heeft dezelfde naam als de `enum` in enkelvoud.	`FieldStyles`
Variabelen, Velden, Attributen	Attributen zijn private.	Publieke variabelen zijn toegestaan. Variabelen die `private` zijn starten niet met een hoofdletter. Namen van variabelen gedeclareerd in een functie mogen in de context begrijpbare afkortingen zijn.	`field` `p` `exampleVar`

Tabel 3.1 Algemene naamgeving entiteiten

Functies die Windows Forms™ gebeurtenissen afhandelen en `private` gedeclareerd zijn, moeten hun door Visual Studio automatisch gegenereerde naam houden (zie § 3.3.2).

3.3.2 Naamgeving Windows Forms Controllers

De gebruikersinterface moet functioneren op basis van de Windows Forms™ technologie van Microsoft Corporation.

Voor de naamgeving van instanties van controllers (superklassen van `Control`) en vensters (superklassen van `Form`) alsmede voor de naamgeving van vensterklassen gelden aparte eisen.

Instantienamen van dergelijke controllers starten met een speciale prefix indien een betreffende prefix in dit document gedefinieerd is. Meestal is dit een afkorting van de oorspronkelijke klassennaam. Niet voor alle controllers is een dergelijke afkorting gedefinieerd, hiervoor geldt de speciale naamgeving niet.

Er moet altijd voldaan worden aan de algemene naamgeving. De prefixen mogen met een hoofdletter starten wanneer dit elders in dit document vereist wordt.

Controller	Prefix	Controller	Prefix
Button	cmd	CheckBox	chk
CheckedListBox, ListBox	lb	ComboBox	cmb
DateTimePicker	dt	Label, LinkedLabel	lbl
ListView	lv	TextBox, MaskedTextbox, RichTextBox	txt

NumericUpDown	ud	PictureBox	pb
ProgressBar	< geen >	RadioButton	rd
TreeView	tv	GroupBox	grp
Panel	pnl	TabControl	< geen >
TabPage	tab	Form	frm

Tabel 3.2 Prefixen voor controllers

3.3.3 Naamgeving attributen, excepties en 'delegates'

Namen van zelf gedefinieerde attributen eindigen altijd met `Attribute`, zoals bijvoorbeeld `SerializableAttribute`. Namen van zelf gedefinieerde exceptieklassen dienen altijd te eindigen met `Exception`, zoals bijvoorbeeld `ArithmeticException`. Namen van instanties van `private` exceptieklassen beginnen met `ex`, eventueel gevolgd door een toevoeging startende met een hoofdletter.

Delegates zijn functiepointers. De naam van een delegate (gedeclareerd met `delegate`) eindigt altijd op `Delegate`, zoals bijvoorbeeld `TestFuctionDelegate`. Merk op dat namen van instanties van delegates niet met `Delegate` hoeven te eindigen.

3.4 Algemene opbouw

3.4.1 Protectieniveaus

Alle C#-entiteiten (waarvoor van toepassing) dienen altijd gedeclareerd te worden met een protectieniveau: `public`, `internal`, `protected`, `protectedinternal` of `private`. Een entiteit mag nooit een minder protectieniveau krijgen, wanneer zonder verdere modificatie aan enige code een zwaarder protectieniveau volstaat. In interne revisies of testversies kan hiervan afgeweken worden.

Een uitzondering hierop zijn lokale variabelen welke `private` zijn, deze krijgen nooit een expliciet protectieniveau.

Wanneer achtereenvolgens meerdere variabelen gedeclareerd worden, worden zij, indien van toepassing, gerangschikt van laag protectieniveau naar hoog protectieniveau. Variabelen gedeclareerd in de blokken `class` en `struct` worden altijd voorgaande aan andere code gedeclareerd.

Publieke velden zijn niet toegestaan als deze niet noodzakelijk zijn voor XML-serialisatie (zie documentatie van .NET Framework). Alhoewel velden gedeclareerd als `internal` zijn toegestaan, dient het gebruikt hiervan beperkt te worden. Wanneer `public` of `internal` velden noodzakelijk zijn, kunnen 'properties' gebruikt worden (zie documentatie van C#). Hieronder een voorbeeld:

```
Color color1;
public Color Color1
{
      get { return color1; }
      set { color1 = value; }
}
```

3.4.2 Commentaar

Commentaar is altijd geschreven in het Nederlands. Zinnen eindigen altijd met een punt. Commentaar is kort en krachtig. Overvloedig commentaar is niet toegestaan, dit verslechtert de leesbaarheid. Dergelijke documentatie kan beter in een ander bestand opgenomen worden.

XML-commentaar is XML code opgenomen in valide C#-commentaar. Dergelijk commentaar moet toegepast worden op entiteiten. Variabelen gedeclareerd in functies en andere `private` variabelen krijgen nooit een dergelijk commentaar. Voor functies die Windows Forms™ gebeurtenissen

afhandelen en `private` gedeclareerd zijn, moet alleen het XML-element `<summary>` opgenomen worden. Dit verbetert de leesbaarheid door redundant commentaar weg te laten.

XML-commentaar moet altijd toegevoegd worden via de ingebouwde functionaliteit van Visual Studio. Dit kan door boven een (ondersteunde) declaratie `///` te typen. Tussen commentaar en declaratie zijn nooit witregels.

Wanneer in een `class` of `struct` variabelen met `private` gedeclareerd worden, dan worden deze voorzien van een korte beschrijving. Deze beschrijving staat achter de declaratie (of achter declaratie + initialisatie). Beschrijvingen van meerdere variabelen onder elkaar worden uitgelijnd door middel van tabs. Zie hieronder een voorbeeld:

```
int xPosition;          //X-Positie van een coördinaat.
int yPosition = 1;      //Y-Positie van een coördinaat.
```

Het taalkundig onderwerp van een beschrijving van een functie is impliciet altijd de functie zelf. Dit onderwerp wordt echter niet expliciet geschreven. Bijvoorbeeld: 'Tekent het scherm', 'Schrijft de instellingen naar bestand file'. Dit is slechts een aanbeveling. Beschrijvingen van constructors beginnen altijd met 'Constructor. Initialiseert de klasse'. Een eventuele toevoeging is toegestaan. XML-Commentaar voldoet bij voorkeur aan de schrijfstijl zoals in de .NET Framework Reference van Microsoft Corporation, hetzij wel in het Nederlands (zie http://msdn.microsoft.com/nl-nl/library/ms229335(en-us).aspx).

3.4.3 Opbouw functies

Functies welke variabelen declareren starten altijd met het commentaar `Init.` gevolgd door variabele declaraties waarbij nooit een expliciet protectieniveau is opgegeven. Voor functies korter dan 10 regels (tellende vanaf het begin van de declaratie) mag `Init.` weggelaten worden. Lokale variabelen mogen overal gedeclareerd worden, echter wanneer mogelijk, moeten zij bovenaan de functie gedeclareerd worden. Bij deze declaraties mag direct een waarde toegekend worden.

Alle logische hoofdgedeeltes van een functie dienen vooraf te gaan aan een korte commentaarregel. Dit kan bijvoorbeeld het begin van een `for`-lus zijn.

Indien een functie resources moet teruggeven of een `return` heeft op het eind van de functie, dan moet dit voorafgegaan worden door het commentaar `Finish.` indien de functie ook een `Init.` commentaar heeft zoals hierboven aangegeven.

De lengte van een functie beginnend vanaf het begin van de declaratie mag nooit langer zijn dan 50 regels. Een lange functie kan opgesplitst worden. Wanneer de functie geen groot algoritme beschrijft, maar wel herhaaldelijke losstaande delen code bevat, mag de functie langer zijn dan 50 regels, doch niet meer dan 100.

3.4.4 Statements

Alle ondersteunende statements van de C#-specificatie versie 2.0 mogen gebruikt worden. Ieder statement staat bij voorkeur op een aparte regel. Korte statements mogen, indien het de leesbaarheid verbeterd, achter elkaar geschreven worden.

Code dient altijd ingesprongen te zijn, in Visual Studio kan dit via 'Edit -> Advanced -> Format Document'.

Een iteratievariabele is een variabele welke in een `for`-lus automatisch een nieuwe waarde krijgt bij iedere iteratie. De namen van dergelijke variabelen zijn altijd opeenvolgend, `i`, `j`, `k`, `l` Waarbij `j` alleen gebruikt mag worden als `i` al in gebruik is, enzovoorts. Wanneer het de context verduidelijkt mag gebruik gemaakt worden van de iteratievariabelen `x`, `y` en `z`.

3.5 Overig beleid

Hieronder volgt een korte opsomming van overige regels waaraan de C#-code moet voldoen:

- Gebruik van het `goto`-statement is verboden.
- Constructors en finalizers worden in een klasse altijd voorgaande aan andere functies opgenomen.

4 Etaris C#-oplossing

De software Etaris is een Tetris variant. Zie voor meer informatie andere projectendocumenten zoals het ´User Requirements Document´ of het ´Software Requirements Document´. Dit document eist een speciale opbouw voor de gehele Visual Studio oplossing van Etaris.

4.1 Pakketten

Etaris bestaat uit een aantal naamruimtes wat in feite pakketen op het ADD niveau zijn, de onderverdeling in de C#-oplossing is lichtelijk anders gekozen.

C#-projecten	Naamruimtes	Type	
Etaris.Gui	Etaris.Gui	Windows applicatie	Naamruimte voor de gebruikersinterface.
Etaris.Core	Etaris.Core	Bibliotheek	Naamruimte welke de volledige spellogica bevat bovenop de definitie van de wereld.
Etaris.Fields	Etaris.Fields	Bibliotheek	Basis naamruimte welke de wereld (in feite de speelvelden) vastlegt. Deze speelvelden bevatten geen automatisch spelgedrag en geen spellogica. De wereld wordt aangestuurd door Etaris.Core.
CommonLibEtaris	syscore syscore	Bibliotheek	Abstracte uitbreiding van .NET (wrappers).

Tabel 4.1 Overzicht C#-oplossing en naamruimtes

Opmerking: `Etaris.Fields` implementeert beweging van speelstukken, maar heeft geen notie van zwaartekracht en weet ook niet wanneer rijen verwijderd mogen worden of wanneer een spel game-over is. Dergelijke zaken worden bepaald door `Etaris.Core`, welke overigens wel de methodes en informatie in `Etaris.Fields` gebruikt.

4.2 Kenmerken C#-oplossingen

Bibliotheken kunnen nooit zelfstandig uitgevoerd worden. Om functies te testen is in iedere bibliotheek (met uitzondering van *CommonLibEtaris*) een *Program.cs* bestand opgenomen. Met dit bestand kan het project opstarten wanneer in de projectinstellingen ‘Console Application’ is ingesteld bij ‘Output type’. Merk op dat wanneer een dergelijke bibliotheek op SVN wordt opgeslagen, dat het ‘Output type’ altijd ‘Class Library’ is.

Voor alle C#-projecten is de hoofd naamruimte ‘Etaris’. Codebestanden welke bij een bepaalde naamruimte horen, worden altijd opgeslagen in een submap met dezelfde naam als de naamruimte (exclusief ‘Etaris.’). Wanneer bestanden in de hoofd naamruimte worden opgeslagen, dan moeten zij in de submap Core worden opgeslagen.

5 Documentinspectie

5.1 Standaard procedure inspectie

Controle van een document geschiedt altijd door de kwaliteitsmanager in samenwerking met minstens één van de betreffende auteurs en een ander persoon welke geen auteur van het betreffende document is. In bijzondere gevallen kan dit geschieden via e-mail.

Opmerkingen dienen bij voorkeur genoteerd te worden in een apart bestand, zodat de auteurs de documenten kunnen wijzigen.

5.2 Bijwerken niet-codebestanden

Een document heeft altijd een eindverantwoordelijke. Dit houdt in dat deze persoon, doch in democratisch overleg, bepaalt wat in een document gewijzigd wordt. De naam van deze persoon staat rechtsonder iedere pagina (exclusief de titelpagina).

Documenten op het SVN systeem (zie Software Configuratie Management Plan) kunnen op diverse manieren veranderd worden:

- Door de eindverantwoordelijke van het document, doorgaans de (hoofd)auteur.
- Door een van de auteurs (eventueel kan ook gebruikt gemaakt worden van redigeren).
- Door een ander persoon. In dit geval moet van redigeren gebruikt gemaakt worden (indien beschikbaar). Dergelijke wijzigingen kunnen ook via e-mail geschieden.

De hoofdauteur kan te allen tijde wijzigingen permanent maken.

Dit beleid is noodzakelijk zodat een eindverantwoordelijke indien gewenst alle wijzingen, losstaand van elkaar, kan accepteren, wijziging of weigeren.

In Microsoft Office-programma's kunnen wijzingen bijgehouden worden via 'Track Changes' of 'Wijzingen bijhouden'. Merk op dat het ook mogelijk is om commentaar te plaatsen via de ingebouwde functionaliteit van Microsoft Office.

5.3 Bijwerken codebestanden

5.3.1 Definities

Zie § 3.1 voor definities welke in onderstaande paragraven gebruikt worden.

5.3.2 Redigeren in codebestanden

Opmerkingen in code kunnen op de volgende manieren worden gemaakt (noot dat het *.reg* bestand in de *template* map op SVN eerst uitgevoerd dient te worden). Dit heeft alleen betrekking op het redigeren, niet op commentaar in zijn algemeen!

```
//HOISUN: Commentaar van Hoisun.
//STELLA: Commentaar van Stella.
//ROAUL: Commentaar van Raoul.
//RICK: Commentaar van Rick.
//ALWIN: Commentaar van Alwin.
//ROLAND: Commentaar van Roland.
```

Dit commentaar is vervolgens zichtbaar in het 'Task List' venster van Visual Studio. Let wel, 'User Tasks' moet naar 'Comments' veranderd worden.

5.3.3 Wijzigingsbeleid codebestanden

Deze paragraaf geeft beleid omtrent het wijzigen van code. Commentaar in de vorm van redigeren zoals hierboven beschreven, wordt in deze paragraaf niet beschouwd als het wijzigen van code.

Een subgroep mag altijd wijzigingen aanbrengen aan zijn codegedeelte, behoudens de volgende gevallen:

- De kwaliteitmanager heeft het wijzigen van bepaalde delen van de code verboden.
- De wijzigingen zijn in strijd met het ADD.
- Wijzigingen hebben grote impact op een codegedeelte van een andere subgroep welke geen toestemming gegeven heeft voor deze wijziging.

De kwaliteitmanager zal altijd de kwaliteit van de code inspecteren. De kwaliteitmanager mag altijd wijzigingen aanbrengen om de algemene kwaliteit te verbeteren.

Indien een subgroep wenst om stubfuncties te maken in een codegedeelte dat niet van haar is, mogen dergelijk functies aangemaakt worden behoudens de volgende gevallen:

- De wijziging is in strijd met de regels voor het wijzingen van eigen code.
- Een andere subgroep wenst dergelijke wijzigingen niet.
- De stubfunctie niet noodzakelijk is voor de voortang van de subgroep.

In een stubfunctie moet het commentaar `//TODO` opgenomen worden. Echter, wanneer een subgroep in haar codegedeelte zelf dergelijke gedeeltelijke functies maakt, hoeft hier niet dit commentaar opgenomen te worden.

Bijlagen

Bijlage A, C#-entiteiten

Hieronder volgt een overzicht van de entiteiten in C# en onder welke subgroep zij vallen.

Categorie	Type	Opmerkingen
Blokken	`namespace`	
	`class`	
	`struct`	
	`interface`	(heeft alleen definities)
	`enum`	
Functies	methods	(in feite normale functies)
	constructors	
	properties (`get`; `set`)	
	`operator`	(handmatig gedefinieerde operatoren)
	indexers	
Velden / variabelen	`event`	(niet als variabele geïmplementeerd)
	`delegate`	
	Overige types	
Overig	`const`	
	Attributes	(is een klasse, met als basisklas `Attribute`)
	Excepties	(is een klasse, met als basisklas `Exception`)

Opmerking: velden zijn variabelen. Echter, zij zijn direct gedeclareerd in een `class` of `struct`.