PSE Treffen Wahlverfahren vom 16.11.2016

-Weitere Vorstellung des Themas (siehe Blatt Aufgabenbeschreibung)

-Q:Warum gibt es in Beispiel 2 unterschiedliche Assertions? (assert.h und das Makro Assert)
A: Man kann sich noch eine Textausgabe ausgeben lassen. Dafür ist das da. Muss man aber nicht verwenden

-Q:Werden die echten Programme früher oder später auf echten Skalen verwendet?
A: Nein. Es werden um einiges kleinere Zahlen verwendet

-Q: Hat der C-Code immer das gleiche Layout?
A:Es gibt eine bestimmte begrenzte Anzahl von Grundlayouts:
1. Normale Mehrheitswahl
2. Zustimmungswahl
3. Präferenzwahl
4. Gewichtete Wahl
Man wählt welches Grundlayout man verwenden will.
Als extra-Feature eventuell zusätzliche Wahlverfahren erlauben

-Q: Soll die GUI ein einzelnes Fenster sein oder mehrere Fenster?
A: Bleibt uns überlassen

Q: Soll alles zusammen gespeichert werden oder einzelne C-Programme sowie Wahlverfahren einzeln gespeichert
A: Es soll auch getrennt möglich sein

Q: Auch UML schon in der 1. Phase
A: Nein! Kein UML in der ersten Phase!

Q: Gibt es globale Testfälle, die wir verwenden können?
A: Wir sollen selbst sinnvolle wählen.

Q: Wie sollen wir speichern? Immer sofort überschreiben?
A: An anderen Texteditoren orientieren. Save & Save as

Q: Wie sollen wir CBMC aufrufen? Die Binary aufrufen? Über C-Code, der dann von Java aufgerufen wird?
A: Es soll auf allen Systemen laufen. User soll Pfad angeben können für den „Windows-fix“
Nativ den C-Code aufrufen ist vermutlich zu schwierig.

Q: Antlr verwenden?
A: Gute Idee. Nicht den C-Parser selbst schreiben!
Darüber nachdenken, welche Version von Antlr wir verwenden wollen.

Zum 2. Editor:
Q: Muss man auch selber Makros erstellen können?
A: Kann man machen muss man aber nicht.

Q: Wenn wir nach dem Pflichtenheft etwas ändern wollen… darf man das?
A: Jein. Wir sollen uns an das Pflichtenheft halten. Wir sollen aber auch keinen „Schrott“ programmieren. Jede Änderung muss begründet und protokolliert werden.

Zu Punkt 4 des Aufgabenblatts:
Wir sollen ein Timeout implementieren, welches CBMC killen kann.
Max-Zeit bzw. Abbrechzeit soll angegeben werden können.

Q: Sollen wir auf CBMC warten oder soll unser Programm während CBMC läuft interaktiv sein?
A: Ist uns überlassen. Schauen, was für den Nutzer sinnvoll ist.

Zu 5.
Die Darstellung ist komplett uns überlassen. Gegenbeispiele mit konkreten Zahlen. Kann aber auch mehr sein.

Q: Wer ist unsere Zielgruppe?
A: Entwickler oder Wahlprüfstelle

Q: Welche Sprachen soll unser Programm unterstützen?
A: Englisch und wenn wir wollen eventuell eine andere Sprache zusätzlich.

Gute Kapselung! Patterns verwenden!

Q: Kann man wissen, wie lange CBMC circa läuft für gewisse Eingaben?
A: Nein. Es wird selbst für mittelgroße Werte viel berechnen müssen. Schwankt natürlich je nach Rechner und Datenstruktur. Es hängt davon ab, wie man das Programm schreibt.

For ist einfacher als while für CBMC

Q: Dürfen wir das SVN für alles verwenden?
A: Ja aber es muss klar sein, was alles bewertet werden soll. Gute Ordnergruppierung!

Auf Lizenzen achten, wenn wir Github verwenden!

Unit Tests sollen mit J Unit geschrieben werden.

Checkstyle von der Programmiervorlesung verwenden (darauf haben wir uns als Team geeinigt)

Generell an Standard-Editoren orientieren was die Features angeht.

Kommentar- und Codesprache konsistent in einer Sprache. Wir einigten uns auf Englisch.

Exceptions fangen! Usabillity optimieren!

Nächstes Treffen Mittwoch 11:30 @50:34 201