PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

Instituto de Ciências Exatas e Informática (ICEI)

Curso de Sistemas de Informação – Processos e Qualidade de Software

Relatório de Análise de Métricas de Qualidade por Programador

Nomes: Leonardo Willian e Marlon Dias

Contextualização

Decidimos, a partir deste trabalho, analisar algumas métricas de qualidade sobre códigos desenvolvidos por contribuintes recentes do projeto open-source da Microsoft, Visual Studio Code, onde avaliamos algum arquivo de código de um repositório pessoal, ou seja, não iniciado por um *fork*. Decidimos então, inicialmente, avaliar as métricas de Padrões de Projeto, Comentários, Indentação e Nomeação das Variáveis.

Ao verificarmos por alto os códigos selecionados, que serão posteriormente aqui citados, percebemos que a métrica de padrões de projeto não poderiam ser aplicadas de forma uniforme, uma vez que em poucos códigos pudemos ver o desenvolvimento de uma classe de forma convencional, enquanto na maioria das vezes, eram utilizadas funcionalidades de bibliotecas já prontas e referenciadas pelo repositório, as quais não poderíamos avaliar, e também sairiam do escopo do trabalho. Por fim, ficamos com as métricas de Comentários, Indentação e Nomeação das Variáveis.

Explicando e pontuando métricas

A nível de explorar as métricas utilizadas no trabalho, é necessário explicar como avaliamos cada uma, haja vista que foi uma análise feita a partir da leitura e “ticagem” das métricas para cada código. Segue abaixo a motivação de avaliação de cada métrica.

• Comentários: há uma discussão muito grande sobre uso de comentários ou não nos códigos, mas decidimos seguir a abordagem descrita pelo livro Código Limpo, onde os comentários devem ser mais sutis, e informar apenas o que é necessário. Vale lembrar que um código bem escrito dificilmente precisará de comentários para explicá-lo. Apesar disso, penalizamos códigos que não continham nenhum comentário. Poderia ser avaliado em 1 – cometeu vacilos mas fez bom uso em outros pontos e 2 – utilizou de forma bem sucinta em todos os pontos;

• Indentação: um código bem indentado é muito mais legível, logo avaliamos esse critério de forma bem sucinta: se indentou de forma correta, ganha o ponto, caso contrário, é penalizado. Foi avaliada em: 0 – Mal indentado ou não indentado e 2 – Corretamente indentado;

• Nomeação das variáveis: Talvez esse seja o aspecto mais importante na legibilidade do código. Uma variável que tenha um nome que represente claramente o que ela armazena, não precisa de mais detalhes para se explicar. Nesse caso, como atribuímos uma pontuação máxima maior, avaliamos em 1 ponto – nomeação pouco clara, 2 – nomeação entendível e 3 – nomeação bem clara.

Programadores e respectivos códigos analisados

Tyriar - https://github.com/Tyriar/vscode-terminal-api-example/blob/master/src/extension.ts
isidorn - https://github.com/isidorn/aspnet/blob/master/Controllers/AccountController.cs
rebornix - https://github.com/rebornix/monaco-vue/blob/master/src/monaco.contribution.ts
sandy081-https://github.com/sandy081/vscode-todotasks/blob/master/src/TodoCommands.ts
bpasero - https://github.com/bpasero/fsevents/blob/master/fsevents.cc
aeschli - https://github.com/aeschli/sample-languages/blob/master/py/python.py
jrieken - https://github.com/jrieken/md-navigate/blob/master/src/extension.ts
ramya-rao-a - https://github.com/ramya-rao-a/electron-crash-server-go/blob/master/main.go
joaomoreno -

https://github.com/joaomoreno/large-scale-typescript/blob/gh-pages/reveal.js/js/reveal.js
egamma - https://github.com/egamma/mobiletry/blob/master/typings/node/node.d.ts

Tyriar em seu código não fez nenhum uso de comentários, logo, não recebeu a ticagem respectiva, indentou corretamente todo o código e as variáveis possuíam nomes que representavam claramente o que armazenavam. Pontuação: 0 + 2 + 3 = 5 pontos

Isidorn fez alguns bons usos de comentários em seu código, mas manteve, como print abaixo, trechos de código comentados, e sem finalidade, o que não é uma prática interessante, o que o penalizou em metade dos pontos de comentários. Fez uma indentação correta, bem como suas variáveis eram bem claras. Pontuação: 1 + 2 + 3 = 6 pontos

```
        [HttpGet]
        [AllowAnonymous]
        public IActionResult ForgotPassword()
        {
            return View();
        }

        //
        // POST: /Account/ForgotPassword
        [HttpPost]
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> ForgotPassword(ForgotPasswordViewModel model)
        {
            if (ModelState.IsValid)
            {
                var user = await UserManager.FindByNameAsync(model.Email);
                if (user == null || !(await UserManager.IsEmailConfirmedAsync(user)))
                {
                    // Don't reveal that the user does not exist or is not confirmed
                    return View("ForgotPasswordConfirmation");
                }

                // For more information on how to enable account confirmation and password reset please visit http://go.microsoft.
                // Send an email with this link
                // var code = await UserManager.GeneratePasswordResetTokenAsync(user);
                // var callbackUrl = Url.Action("ResetPassword", "Account", new { userId = user.Id, code = code }, protocol: Conte
                // await MessageServices.SendEmailAsync(model.Email, "Reset Password",
                //    "Please reset your password by clicking here: <a href=\"" + callbackUrl + "\">link</a>");
                // return View("ForgotPasswordConfirmation");
            }

            // If we got this far, something failed, redisplay form
            return View(model);
```

Rebornix foi bem claro nos três pontos analisados, portanto recebeu a pontuação máxima: 7 pontos

Sandy081 não fez uso nenhum de comentários, portanto foi penalizado nessa métrica, porém fez uma indentação correta e suas variáveis eram bem claras. Pontuação: 0 + 2 + 3 = 5 pontos

Bpasero fez comentários sucintos em seu código, bem como indentou corretamente, porém, suas variáveis foram pouco claras, e houve um caso, conforme print abaixo, onde a variável foi nomeada como “tp1” e isso não representou nada naquele código.

```
void FSEvents::Initialize(v8::Handle<v8::Object> exports) {
  v8::Local<v8::FunctionTemplate> tpl = Nan::New<v8::FunctionTemplate>(FSEvents::New);
  tpl->SetClassName(Nan::New<v8::String>("FSEvents").ToLocalChecked());
  tpl->InstanceTemplate()->SetInternalFieldCount(1);
  tpl->PrototypeTemplate()->Set(
           Nan::New<v8::String>("start").ToLocalChecked(),
           Nan::New<v8::FunctionTemplate>(FSEvents::Start));
  tpl->PrototypeTemplate()->Set(
           Nan::New<v8::String>("stop").ToLocalChecked(),
           Nan::New<v8::FunctionTemplate>(FSEvents::Stop));
  exports->Set(Nan::New<v8::String>("Constants").ToLocalChecked(), Constants());
  exports->Set(Nan::New<v8::String>("FSEvents").ToLocalChecked(),
               tpl->GetFunction());
}

NODE_MODULE(fse, FSEvents::Initialize)
```

Aeschli cumpriu com todos os aspectos analisados, e também recebeu pontuação máxima: 7 pontos

Jrieken não fez uso de comentários, porém indentou corretamente e suas variáveis foram bem claras. Pontuação: 0 + 2 + 3 = 5 pontos

Nenhum comentário explicando o que a função faz ou algo do tipo:

```
1   'use strict';
2
3   import * as vscode from 'vscode';
4
5   export function activate(context: vscode.ExtensionContext) {
6       const registration = registerDocumentSymbolProvider();
7       context.subscriptions.push(registration);
8   }
9
10  function registerDocumentSymbolProvider(): vscode.Disposable {
11
12      const _atxPattern = /^(#){1,6}\s+.+/;
13      const _settext = /^\s*[-=]+\s*$/;
14
```

Ramya-rao-a não fez uso de comentários, indentou corretamente o código, e suas variávels tiveram nomes razoáveis, porém, não totalmente claros como em um exemplo onde uma variável foi nomeada com a letra “p”. Pontuação: 0 + 2 + 1,5 = 3,5 pontos

Nenhum comentário explicando a função e uso de variáveis nada descritivas:

```
func viewHandler(w http.ResponseWriter, r *http.Request, title string) {
        p, err := loadPage(title)
        if err != nil {
                http.Redirect(w, r, "/edit/"+title, http.StatusFound)
                return
        }
        renderTemplate(w, "view", p)
}
```

Joaomoreno aplicou bem todas as métricas analisadas, recebendo portanto, pontuação máxima: 7 pontos

Egamma, por fim, não utilizou de comentários, indentou corretamente e suas nomeações de variáveis foram razoáveis, porém, não totalmente claras. Pontuação: 0 + 2 + 1,5 = 3,5 pontos

Ausência de explicação da Interface e algumas variáveis com abreviações não explicativas:

```
17
18  interface NodeProcess extends NodeEventEmitter {
19      stdout: WritableStream;
20      stderr: WritableStream;
21      stdin: ReadableStream;
22      argv: string[];
23      execPath: string;
24      abort(): void;
25      chdir(directory: string): void;
26      cwd(): string;
27      env: any;
```

Para avaliarmos os comentários, seguimos algumas orientações descritas no link abaixo:

http://www.itnerante.com.br/profiles/blogs/clean-code-3-coment-rios