دانشکده مهندسی
کامپیوتر و فناوری اطلاعات

1399/10/07

تمرین پنجم

مهندسی نرم افزار 2

گروه {#}

اعضاء گروه:

1) علی خرمی‌پور
2) امیررضا شیرمست
3) علیرضا صدیقی‌مقدم

1) درباره رهنمودهای ساخت نرم افزار قابل نگهداری و معیارهای ارزیابی قابلیت نگهداری، که توسط Software Improvement Group (SIG) معرفی شده است، مطالعه کنید[1].

الف) رهنمودهای مطرح شده را مختصراً معرفی کرده و (بر اساس نظر و تجربه خود) اولویت بندی کنید.

رهنمودهای مطرح‌شده بر اساس اولویت:

1. **Write clean code**

کد تمیز، کدی قابل نگه‌داری است؛ برای همین تمیز کد نوشتن قابلیت نگه‌داری نرم‌افزار را افزایش می‌دهد. هم‌چنین باید از "code smell"ها پرهیز کرد، چرا که نشان‌دهنده‌ی وجود مشکل هستند.

2. **Write simple units of code**

اگر قسمت‌هایی که کد چند شاخه می‌شود(به عنوان مثال عبارات شرطی) حداقل شوند، تغییر و تست کد راحت‌تر می‌شود؛ برای این کار می‌توان قسمت‌های پیچیده‌تر را به قسمت‌های ساده‌تر تقسیم کرد. نکته‌ای که در کتاب گفته‌شده این است که تعداد دفعات چندشاخه شدن کد به ازای هر واحد حداکثر ۴ باشد.

3. **Write short units of code**

خواندن، تست و استفاده مجدد واحدهای کوتاه‌تر کد، راحت‌تر است و قابلیت نگه‌داری را افزایش می‌دهد. بهتر است که هر واحد کد دارای حداکثر ۱۵ خط کد باشد.

4. **Write code once**

در صورتی که کد کپی شود و در آن اشکالی وجود داشته‌باشد، اشکالات باید در چند نقطه رفع شوند؛ این باعث می‌شود که قابلیت نگه‌داری کاهش‌یابد. می‌توان با استفاده از متدها یا استفاده از کد قابل استفاده مجدد، از کپی‌کردن کد جلوگیری کرد.

5. **Keep unit interfaces small**

با کم‌کردن تعداد پارامترهای هر واحد کد، فهم کد راحت‌تر شده و قابلیت استفاده مجدد فراهم می‌شود که منجر به افزایش قابلیت نگه‌داری می‌شود. بهتر است حداکثر تعداد پارامترهای هر واحد ۴ باشد.

[1] https://cutt.ly/Jh8yxTA

6. **Separate concerns in modules**

می‌توان با عدم استفاده از ماژول‌های بزرگ، از طریق تخصیص مسئولیت‌های متفاوت به ماژول‌های جدا و پنهان‌کردن پیاده‌سازی در پشت واسط‌ها، به coupling کم دست‌یافت. این کار باعث افزایش قابلیت نگه‌داری می‌شود، چرا که نظارت بر و اجرای تغییرات در یک سیستم با coupling کم راحت‌تر خواهد بود.

7. **Couple architecture components loosely**

باید بین اجزای معماری coupling کمی باشد؛ چرا که باعث تسهیل در نگه‌داری از جز می‌شود. می‌توان با حداقل‌کردن ارتباط بین ماژول‌ها (فراخوانی یک قسمت از کد یک ماژول توسط دیگری) به هدف مذکور رسید.

8. **Keep your codebase small**

باید اندازه کد سیستم حداقل باشد؛ چرا که داشتن یک محصول، یک پروژه و یک تیم کوچک منجر به افزایش قابلیت نگه‌داری می‌شود. باید به‌صورت فعال سعی کرد تا سایز سیستم کاهش یابد.

9. **Keep architecture components balanced**

با یکسان‌سازی اندازه اجزای معماری، نگه‌داری افزایش می‌یابد؛ چرا که باعث تسهیل پیداکردن کد موردنظر و نگه‌داری آن به جز به‌صورت مجزا می‌شود. طبق راهنمایی کتاب، بهتر است تا تعداد اجزا نزدیک به ۹ باشد (عددی بین ۶ و ۱۲)؛ هم‌چنین بهتر است تا اندازه اجزا به یکدیگر نزدیک باشد.

10. **Automate tests**

با خودکار کردن تست‌ها، فرایند توسعه قابل پیش‌بینی شده و ریسک آن کاهش می‌یابد که منجر به افزایش قابلیت نگه‌داری می‌شود.

ب) قابلیت نگهداری یک پروژه دلخواه را با استفاده از روش پیشنهاد شده این گروه بررسی کنید. نتایج به دست آمده را تحلیل کنید.

2) مقاله "Architectural Mismatch: Why Reuse Is Still so Hard"[2] را مطالعه کنید.

الف) به نظر شما چالش‌های مطرح شده در این مقاله بر طرف شده‌اند یا همچنان معتبر هستند؟ توضیح دهید.

در سال ۱۹۹۵ مقاله‌ای با عنوان "Architectural Mismatch: Why Reuse Is So Hard" چاپ شد که در آن به معضلات استفاده‌ی مجدد به دلیل ناسازگاری معماری پرداخته‌شد. مفروضاتی که ممکن بود باعث ناسازگاری معماری شوند به ۴ دسته‌ی اصلی تقسیم شده‌بودند:

- ساختار اجزا
- ساختار ارتباط‌ها
- ساختار کلی معماری
- فرایند ساخت (محیط توسعه و ساخت)

هم‌چنین، سه وجه از تعامل بین اجزا که ممکن است منجر به ناسازگاری شوند عبارت‌اند از:

- زیرساختی که جز بر روی آن متکی است
- نرم‌افزاری که از جز استفاده می‌کند
- ارتباط بین اجزای هم‌سطح

مقاله "Architectural Mismatch: Why Reuse Is Still So Hard" ۱۴ سال بعد از مقاله‌ی اول، در سال ۲۰۰۹ منتشر شد و سعی داشت تا به همان مشکلات مقاله‌ی اول بپردازد و بررسی کند که آیا گذر زمان از مشکلات کاسته‌است یا خیر.

برای دست و پنجه نرم‌کردن با ناسازگاری می‌توان در وهله‌ی اول از آن پیشگیری کرد؛ در صورت عدم امکان باید بتوان آن را شناسایی و در قدم آخر و در صورت اجتناب‌ناپذیری، باید آن را حل نمود. روش‌های توسعه‌ی نرم‌افزار در هر سه زمینه پیشرفت‌هایی داشته‌اند؛ اما نه تنها معضلات مطرح‌شده در مقاله قدیمی‌تر هنوز هم وجود دارند، بلکه مشکلات دیگری هم اضافه شده‌اند از قبیل:

- اعتماد
- پویایی (Dynamism)
- تکامل معماری
- قفل‌شدن معماری (Architecture Lock-In)

در آخر، تمام این معضلات باقی مانده‌اند و حتی به دلیل افزایش استفاده مجدد و افزایش پیچیدگی مفروضات، جدی‌تر نیز شده‌اند که نیازمند توجه بیشتر مهندسین نرم‌افزار خواهدبود.

[2] برای دریافت مقاله به قسمت "مطالعه بیشتر" در مودل مراجعه کنید.

ب) چه پیشرفت‌ها و تغییراتی در حوزه مهندسی نرم افزار (توسعه نرم افزار)، موجب برطرف شدن چالش های قبلی یا ایجاد چالش‌های جدید شده است؟ توضیح دهید.

- اعتماد: یکی از مسائل نرم‌افزارهای سطح اینترنت، اعتماد بین اجزا است. بسیاری از رخنه‌های امنیتی به‌دلیل عدم ایمن‌سازی کافی نرم‌افزار بوده‌اند. با افزایش نرم‌افزارهای سطح اینترنت، این چالش باید بیشتر مورد توجه قرار گیرد.

- پویایی (Dynamism): پیکربندی مجدد پویا (Dynamic reconfiguration) از اهمیت به‌سزایی برخوردار است؛ چرا که امروزه سیستم‌ها باید بیشتر با منابع متغیر، اجزایی که ممکن است دچار اشکال شوند و تغییر نیاز کاربران دست و پنجه نرم کنند. این مسئله باعث می‌شود تا ترمیم mismatch سخت‌تر شود؛ چرا که در حین اجرا باید تنظیمات اعمال شوند.

- قفل‌شدن معماری (Architecture Lock-In): بعد از فراگیر شدن اینترنت و به‌وجود آمدن سرویس‌های ابری، در صورتی که تغییر معماری از قبل پیش‌بینی نشده‌باشد، ممکن است ایجاد تغییر از نظر اقتصادی به صرفه نباشد؛ در نتیجه بسیاری از سرویس‌ها قابل استفاده مجدد نخواهندبود.

3) دو دسته از ابزارهای مدیریت پیکربندی، ابزارهای مدیریت کد منبع و ابزارهای ساخت و ادغام پیوسته هستند.

الف) این دو دسته را مختصراً معرفی کنید.

ابزارهای مدیریت کد منبع وظیفه دنبال کردن و مدیریت تغییرات کد نرم‌افزار را بر عهده دارند. این ابزارها تیم‌های نرم‌افزاری را یاری می‌دهند تا تغییراتی که هرکدام از اعضای تیم در طول زمان ایجاد می‌کنند به صورتی مدیریت کنند که این همکاری موجب کار سریعتر و هوشمندتر شود نه اینکه هماهنگی بین اعضای تیم برای ایجاد تغییرات به یک سرباری برای کار تبدیل شود.

ابزارهای ساخت و ادغام پیوسته وظیفه خودکار سازی امر یکپارچه سازی کدهایی را بر عهده دارد که از چند مبدا دریافت شده و برای محصول نهایی باید در کنار هم قرار گیرند. این ابزارها باعث می‌شوند که توسعه‌دهنده به صورت متناوب تغییرات را در صورت صحت کد و پاس شدن تست‌های خودکار در محصول نهایی اعمال کند.

ب) از هر دسته سه ابزار انتخاب کنید: ضمن معرفی هر ابزار، امکانات هریک را بررسی و با یکدیگر مقایسه کنید.

ابزارهای مدیریت کد منبع:

1. Git:
   - قابلیت‌ها:
     - پشتیبانی قوی از توسعه غیر خطی
     - مدل توزیع شده‌ی repository
     - سازگار با سیستم‌ها و پروتکل‌های موجود مانند HTTP، FTP، SSH و...
     - توانایی کنترل پروژه‌های کوچک و بزرگ
     - احراز هویت رمزگذاری شده
     - قابلیت ادغام منعطف
   - مزایا:
     - کارایی سریع و بهینه
     - قابلیت Cross-platform
     - تغییرات کد به راحتی دنبال می‌شوند
     - قابلیت نگهداری قدرتمند
     - فراهم کردن ابزار هم به صورت خط فرمان(Command line) و هم به صورت رابط گرافیکی(GIT GUI)
   - معایب:
     - Log history پیچیده و سختی فهم آن

- عدم پشتیبانی از keyword expansion و timestamp

2. CVS:

- قابلیت‌ها:
  - مدل client-server repository
  - قابلیت دسترسی ناشناس برای خواندن کد
  - قابلیت پشتیبانی از شاخه‌های مختلف پروژه
  - اعمال سیاست‌های مختلف امنیتی
  - اعمال فشرده سازی برای مدیریت حافظه
- مزایا:
  - قابلیت cross-platform
  - پشتیبانی وسیع
  - ابزاری شناخته شده و جا افتاده
  - قابلیت همکاری به صورت متن باز
- معایب:
  - عدم بررسی یکپارچگی کد
  - عدم پشتیبانی از commit های اتمیک
  - پشتیبانی ضعیف از منابع توزیع شده
  - عدم پشتیبانی از ردیابی ادغام

3. SVN:

- قابلیت‌ها:
  - مدل client-server repository
  - نسخه‌بندی دایرکتوری‌ها
  - نسخه‌بندی اعمال copy، delete، move و rename
  - پشتیبانی از commit های اتمیک
  - نسخه‌بندی metadata
  - بهینه سازی فضای حافظه
- مزایا:
  - ابزارهای رابط کاربری مانند TortoiseSVN

- پشتیبانی از دایرکتوری‌های خالی
- پشتیبانی بهتر از windows در مقایسه با git
- مدیریت آسان
- یکپارچگی مناسب با windows، IDE های معروف و ابزارهای agile

- معایب:
  - عدم ذخیره زمان تغییر فایل‌ها
  - عدم پشتیبانی از نرمال سازی نام فایل‌ها
  - عدم پشتیبانی از signed revisions

ابزارهای ساخت و ادغام پیوسته:

1. Jenkins:
   - نصب و بروزرسانی آسان بر روی سیستم‌عامل های مختلف
   - رابط کاربری ساده و user-friendly
   - قابلیت انعطاف با افزونه‌های منابع community-contributed
   - قابلیت پیکربندی ساده محیط رابط کاربری
   - پشتیبانی از build های توزیع شده با مدل master-slave
   - زمانبندی build ها
   - پشتیبانی از windows command execution
   - پشتیبانی از اعلان وضعیت در هنگام build

2. CircleCI:
   - یکپارچگی با GitHub ،Bitbucker
   - اجرای build ها با استفاده از container و ماشین‌های مجازی
   - Debugging ساده
   - موازی سازی خودکار
   - تست سریع
   - قابلیت شخصی سازی وسیع
   - نصب سریع و build نامحدود

3. TeamCity:

- ارائه چندین روش برای تنظیمات استفاده مجدد
- اجرای موازی build ها به صورت همزمان و بر روی محیط‌های مختلف
- مشاهده تاریخچه تست‌ها، pinning، tagging و اضافه کردن build ها به دسته favorite
- شخصی‌سازی و توسعه آسان سرور
- نگهداری پایدار از سرورهای CI
- مدیریت منعطف کاربرها

4) فعالیت‌های پیکربندی نرم افزار را در حوزه ساخت برنامه های 'نرم افزار به عنوان خدمت'[3] بررسی کنید. مهمترین چالش‌ها و مزایای آن را توضیح دهید.

نرم‌افزار به عنوان خدمت سیستم‌های نرم‌افزاری هستند که برروی سرویس‌های ابری اجرا شده و از طریق اینترنت در اختیار کاربران قرار داده می‌شود و کاربران می‌توانند با آن تعامل داشته‌باشند.

چالش‌ها:

- نیاز به پیکربندی در لایه شبکه برای فراهم کردن کیفیت مناسب
- در صورت تبادل اطلاعات پر حجم میان کاربر و سیستم، هزینه ارتباط بالا خواهد بود
- تأمین امنیت داده‌های کاربر به دلیل اشتراک گذاری آنها بر بستر اینترنت
- تنوع قدرت شبکه‌های کاربران و مشکل برای کاربرانی با اینترنت‌های ضعیف
- قابلیت Scalability بالای سیستم برای مدیریت تعداد زیاد کاربران در واحد زمان

مزایا:

- مشخص بودن سیستم هدف و عدم نیاز به مدیریت سخت‌افزارهای مختلف
- همیشه آخرین نسخه در اختیار کاربر قرار می‌گیرد و نگرانی از بابت عدم تطابق نسخه‌های مختلف یا یکدیگر نداریم
- قابلیت استفاده در هر سیستم متصل به شبکه اینترنت
- عدم نیاز به فراهم کردن سخت‌افزار قدرتمند در سمت کاربر
- کاهش هزینه‌ها

[3] Software as a Service (SaaS)

- پاسخ تمرین ها را به زبان فارسی و به صورت تایپ شده، در قالب یک فایلPdf ، در مودل بارگزاری کنید.
- لطفا نظم، ساختار و توالی سوالات را در پاسخ ها رعایت کنید.
- **در هر پاسخ، منابع استفاده شده را درج نمائید.**
- فایل پاسخ تمرین را تنها با قالب **SE2-HW5-GroupNumber.pdf** در مودل بارگزاری کنید.
- بارگزاری تمرین توسط یکی از اعضاء گروه کافی است.
- برای پاسخ‌های هر قسمت منابع استفاده شده را درج نمائید.
- فایل زیپ ارسال **نکنید**.
- به ازای هر روز تاخیر در تحویل تمرین 20٪ از نمره تمرین کسر خواهد شد.
- حداقل برخورد به پاسخ‌های مشابه، تخصیص نمره کامل منفی به طرفین خواهد بود.