دانشکده مهندسی کامپیوتر

شبکه‌های پیچیده پویا

پروژه اول

سنجش کیفیت معماری نرم‌افزار

مدرس:

دکتر حسین رحمانی

کمک مدرس:

ذاکری ـ ملکی‌فر

۳ اسفند ۱۳۹۸

فهرست مطالب

۱ مقدمه ۱

۲ استخراج الگوهای طراحی ۱

۱.۲ تبدیل کد منبع به گراف ۲

۲.۲ تولید گراف الگوهای طراحی ۳

۳.۲ یافتن الگوهای طراحی در نرم‌افزار ۳

۴.۲ پروژه ـ فاز اول ۴

۳ استخراج معماری ۴

۱.۳ تعیین پیمانه‌ها ۵

۲.۳ پروژه ـ فاز دوم ۵

تحویل پروژه ۶

مراجع ۷

۱ مقدمه

«برنامه‌نویسی هنر گفتن چیزی که یک نفر از کامپیوتر می‌خواهد تا انجام دهد، به انسان دیگری است.»

✠ دونالد کنوث

مصنوعات نرم‌افزاری در مراحل مختلف توسط گراف قابل توصیف هستند. در نتیجه می‌توان از فنون تحلیل شبکه در مهندسی نرم‌افزار استفاده کرد. در این پروژه کیفیت نرم‌افزار را در سطح طراحی با بهره‌گیری از تحلیل‌های گرافی، مورد ارزیابی قرار می‌دهیم. دو تحلیل مورد نظر در اینجا، استخراج خودکار الگوهای طراحی و استخراج معماری نرم‌افزارهای شی‌گرا با پیمانه‌بندی (بازمهندسی) کلاس‌های برنامه است. هدف از این پروژه، آشنایی با کاربرد دانش شبکه و تحلیل گراف، در مهندسی نرم‌افزار است. در بخش ۲، کلیات نحوه استخراج الگوهای طراحی از متن برنامه، شرح داده می‌شود. در بخش ۳، نیز کلیات نحوه استخراج معماری از متن برنامه، مطرح می‌گردد. در پایان هر بخش گام‌های تحویل دادنی پروژه ذکر گردیده است.

۲ استخراج الگوهای طراحی

الگوهای طراحی ابزارهای با ارزشی در تولید نرم‌افزارهای حرفه‌ای می‌باشند، آنها باعث ساده‌تر شدن مراحل طراحی، پیاده‌سازی و نگهداری سیستم‌های نرم‌افزاری می‌شوند.هر چه تعداد الگوهاي بکار رفته شده و کیفیت به‌کارگیری الگوها بیشتر باشد، قابلیت اطمینان، قابلیت درک و توسعه و در مجموعه کیفیت نرم‌افزار بیشتر خواهد بود. هدف سنجش کیفیت نرم‌افزار بر مبناي میزان استفاده بهینه و تطابق کد برنامه‌ها با الگوهای شناخته شده طراحي است. از آنجایي که پیاده‌سازي الگوهای طراحي وابسته به سلیقه و درک برنامه نویس از الگوها متفاوت است، الگوریتم قطعی برای تشخیص الگوها وجود نداشته و استفاده از روش‌های اکتشافی در این زمینه امیدبخش‌تر بوده است. در مرحله اول این پروژه بر روی روش‌های sub-graph matching تمرکز می‌کنیم. برای این منظور ابتدا کد برنامه شی‌گرا را یک نمایش گرافی تبدیل می‌کنیم.

۱.۲ تبدیل کد منبع به گراف

الگوهای طراحی دارای نمودارهای کلاس مشخصی هستند. کد منبع هر برنامه نیز نمودار کلاس مختص به خود را دارد. هر نمودار کلاس در UML را می‌توان توسط یک گراف (جهت‌دار) نشان داد. جهت یال همان جهت استاندارد برای بیان روابط بین کلاس‌ها در UML است. علاوه‌بر این، هر یال یک برچسب دارد که نوع ارتباط را مشخص می‌کند. برچسب یال‌ها ارتباط‌های موجود در نمودار UML هستند. برچسب‌های مورد انتظار برای هریال در اینجا عبارتند از:

I. Generalization: بین کلاس A و کلاس B رابطه وراثت وجود دارد (کلاس B از کلاس A ارث‌بری کرده است).

II. Association: بین کلاس A و کلاس B رابطه انجمنی وجود دارد. جهت این رابطه با توجه به نحوه استفاده نمونه‌های یک کلاس از کلاس دیگر مشخص می‌شود.

III. Aggregation: بین کلاس A و کلاس B رابطه تجمیع وجود دارد. یک یا چند شیء از کلاس B در کلاس A تعریف شده است. نمونه‌های B مستقل از A وجود دارند.

IV. Composition: بین کلاس A و کلاس B رابطه ترکیب وجود دارد. یک یا چند شیء از کلاس B در کلاس A تعریف شده است و نمونه‌های B به نمونه‌های A وابسته‌اند.

ابزارهای متعددی برای مصورسازی کد و استخراج نمودار ارتباطی کلاس‌ها از کد منبع برنامه آنها وجود دارد [۱–۳]. این ابزارها زبان‌های برنامه نویسی متداول را پشتیبانی می‌کنند. اما نمودار کلاس UML بایستی مورد پیش‌پردازش قرار گرفته و گراف ارتباطی کلاس‌ها از آن استخراج شود؛ زیرا، در نمودار کلاس اطلاعات بیش‌تری مانند نوع و نام فیلدها وجود دارد که در اینجا به آنها نیاز نداریم. استفاده از هر ابزاری برای تولید گراف در این قسمت مجاز است. برای نمونه ابزار Enterprise Architect [۱] به آسانی نمودار کلاس را از کد منبع یک برنامه تولید کرده و آن را در قالب XMI که خود نوعی فایل XML است، به‌عنوان خروجی می‌دهد. با خواندن و پردازش این فایل، گراف ارتباطی برنامه قابل ایجاد است. یک نمونه برنامه تجزیه‌گر برای فایل‌های XMI در [۴] آمده است که عملیات استخراج اطلاعات از این فایل را با فراهم آوردن API مناسب تسهیل می‌کند.

۲.۲ تولید گراف الگوهای طراحی

نمایش گرافی ارتباط کلاس‌ها این ویژگی را دارد که مستقل از زبان برنامه‌نویسی است. الگوهای طراحی نیز ذاتاً مستقل از برنامه هستند و گراف آنها را می‌توان از پیاده‌سازی آنها استخراج کرد. پیاده‌سازی جامعی از الگوهای طراحی مهم با زبان C++ در [۵] آمده است. همچنین در همین منبع شرح کامل هر الگوی طراحی، آورده شده است. پیاده‌سازی دیگری نیز در آمده است [۶]. برای هر الگوی طراحی گراف نظیر آن، مشابه بخش ۱.۲، قابل تولید است.

۳.۲ یافتن الگوهای طراحی در نرم‌افزار

هر نرم‌افزار شی‌گرا ممکن است از یک یا تعدادی از الگوهای طراحی استفاده کرده باشد. این الگوهای طراحی استفاده شده به‌صورت زیرگرافی[1] در گراف اصلی برنامه، وجود خواهند داشت. در این صورت با جست‌وجوی گراف مربوط به هر الگوی طراحی در گراف اصلی و تطبیق یک زیرگراف یافت شده با آن الگوی طراحی، استفاده یا عدم استفاده از این الگو در گراف مشخص می‌شود. تشخیص الگوها صرفاً با استفاده از یک‌ریختی گراف[2]،

[1] sub-graph

[2] Graph isomorphism

دقیق نبوده و همان‌طور که گفتیم قابل بهبود است. با این حال در این مرحله هدف استفاده تطبیق گراف الگوهای طراحی با زیرگراف‌هایی از گراف اصلی برنامه و تعیین الگوها است.

۴.۲ پروژه ـ فاز اول

مراحل زیر را انجام داده، ابزارها و گزارش‌های خود را در ارتباط با هر گام تحویل دهید:

۱. گراف ارتباطی کلاس‌ها را برای پیمانه‌های control[۱] و planning[۲] سیستم اتومبیل خودران Apollo [۷]، استخراج نمایید.

۲. ویژگی‌های اولیه گراف، مانند توزیع درجه و گره‌های مرکزی را برای گراف هر کدام از پیمانه‌ها مشخص نمایید. با تحلیل نتایج مهمترین کلاس‌های هر کدام از پیمانه‌ها را تعیین کنید.

۳. گراف ارتباطی کلاس‌ها را برای الگوهای طراحی در [۵]، به‌دست آورید.

۴. الگوهای طراحی استفاده شده در هر کدام از پیمانه‌های control و planning را مشخص و گزارش نمایید.

۵. با تعیین نسبت تعداد کلاس‌هایی که عضوی از یک الگوی طراحی بوده‌اند به تعداد کل کلاس‌های برنامه (پیمانه)، کیفیت آن پیمانه را محاسبه کنید.

۶. نقاط ضعف و قوت روش خود در هر گام را توضیح دهید.

۳ استخراج معماری

مهندسی معکوس کد و استخراج معماری از کد منبع برنامه (به بیان دقیق تر استخراج مدل ارتباطی کلاس ها و تعیین طرح معماری نرم افزار) عمدتاً با سه رویکرد صورت می پذیرد. نخست، تولید مستندات از متن برنامه موجود و از پیش نوشته شده، به منظور نگهداشت و توسعه آن. دوم، کشف ساختار کدهای حجیم به منظور درک

[۱] https://github.com/ApolloAuto/apollo/tree/master/modules/control
[۲] https://github.com/ApolloAuto/apollo/tree/master/modules/planning

عملکرد آن ها و تأثیرشان بر روی سیستم های کامپیوتری و سوم سنجش کیفیت نرم‌افزار. یک روش دیگر برای سنجش کیفیت معماری نرم‌افزار، اندازه‌گیری پیمانگی [۱] است. چسبندگی [۲] و اتصال [۳] دو اصل حاکم بر پیمانه‌ها در نرم‌افزار هستند. در یک پیمانه‌بندی خوب شاهد پایین بودن اتصال و بالا بودن چسبندگی هستیم. این مفاهیم دقیقاً همان مفاهیم مطرح در تشخیص جوامع در شبکه‌های پیچیده هستند. اصول یادشده بر این نکته تأکید دارند که ارتباط بین گره‌ها داخل یک جامعه زیاد و ارتباط بین گره‌ها بین دو جامعه متمایز کم است. بنابراین هدف بسته‌بندی کلاس‌ها داخل پیمانه‌های مختلف است به نحوی که اختلاف چسبندگی و اتصال بیشینه شود.

۱.۳ تعیین پیمانه‌ها

با استخراج گراف ارتباطی کلاس، امکان استخراج پیمانه‌ها از طریق روش‌های خوشه‌بندی گراف و تشخیص جامعه میسر می‌شود. برای این منظور می‌توان از گراف به‌دست آمده در بخش ۲، استفاده کرده و عملیات خوشه‌بندی را بر روی آن انجام داد. الگوریتم‌های مختلفی برای تشخیص جامعه، در حوزه شبکه‌های پیچیده ارائه شده است. می‌توان تعدادی از این الگوریتم ها را استفاده و نتایج را با یکدیگر مقایسه کرد. یک روش و ابزار برای این منظور در [۸، ۹] معرفی شده است.

۲.۳ پروژه ـ فاز دوم

مراحل زیر را انجام داده و ابزارها و گزارش‌های خود را تحویل دهید:

۱. با استفاده از خوشه‌بندی گراف ارتباطی استخراج شده در فاز اول پروژه برای پیمانه‌های control و planning، معماری هرکدام را در سطح نمودار مؤلفه پیدا کرده و میزان پیمانگی را مشخص نمایید.

۲. میزان پیمانگی را برای معماری فعلی هرکدام از پیمانه‌های control و planning، محاسبه کنید. میزان بهبود پیمانگی با روش پیمانه‌بندی خودکار چه‌قدر است؟

[1] modularity

[2] cohesion

[3] coupling

۳. گراف ارتباطی را بدون جهت در نظر بگیرد و گام‌های (۱) و (۲) را انجام دهید. نتایج را با گام (۱) مقایسه کنید.

۴. **(اختیاری)** گراف ارتباطی را به‌صورت جهت‌دار و وزن‌دار در نظر گرفته و مراحل را تکرار کنید. وزن هر یال در این حالت برابر با تعداد ارتباطاتی است که از کلاس اول (فراخواننده) به کلاس دوم (فراخوانده شده) وجود دارد. یک نمونه از این کار در [۸] انجام شده است. نتایج این گام را گام‌های قبلی مقایسه کنید.

۵. **(اختیاری)** برنامه‌ای بنویسید که گراف مؤلفه‌های استخراج شده و کلاس‌های داخل هر یک را به یک فایل XMI تبدیل کند به نحوی که توسط نرم‌افزار Enterprise Architect [۱] باز شود. با این کار، مرحله طراحی در فرایند مهندسی نرم‌افزار را به‌طور کامل خودکار نموده و یکی از مراحل این فرایند را حذف کرده‌اید.

۶. نقاط ضعف و قوت روش خود در هر گام را توضیح دهید.

تحویل پروژه

یک قالب گزارش LaTex از طریق لینک زیر در اختیار شما قرار داده می‌شود که برای هریک از گام‌های موجود در پروژه یک فضا در نظر گرفته است. شما بایستی آن را تکمیل نموده و تحویل دهید. موعد تحویل پروژه نیز اطلاع رسانی خواهد شد. در صورتی که مشکلی در نگارش این سند وجود داشت و یا مطلبی برای شما مبهم بود می توانید از طریق نشانی ایمیل m-zakeri@live.com درخواست و یا پرسش خود را ارسال فرمایید.

◁ دریافت قالب گزارش:

https://www.dropbox.com/s/bwb9yuq4rapv9mx/report_template_project1.zip?dl=0

مراجع

[1] S. S. P. Ltd., "Enterprise architect," [Online]. Available: https://sparxsystems.com/, [Accessed: 2020-01-27].

[2] V. Paradigm, "Visual paradigm," [Online]. Available: https://www.visual-paradigm.com/, [Accessed: 2020-01-27].

[3] I. Scientific Toolworks, "Understand," [Online]. Available: https://scitools.com/, [Accessed: 2020-01-01].

[4] SDMetrics, "The software design metrics tool for the uml," [Online]. Available: https://www.sdmetrics.com/CustomXMI.html, [Accessed: 2020-01-28].

[5] Refactoring.Guru, "Design patterns in C++," [Online]. Available: https://refactoring.guru/design-patterns/cpp, [Accessed: 2020-01-25].

[6] Wikibooks.org, "C++ programming: Code patterns design," [Online]. Available: https://en.wikibooks.org/wiki/C%2B%2B_Programming/Code/Design_Patterns, [Accessed: 2020-01-25].

[7] A. Auto, "Apollo," [Online]. Available: http://apollo.auto/, [Accessed: 2020-01-25].

[8] B. S. Mitchell and S. Mancoridis, "On the automatic modularization of software systems using the bunch tool," *IEEE Transactions on Software Engineering*, vol.32, pp.193–208, March 2006.

[9] S. Mancoridis, B. S. Mitchell, Y. Chen, and E. R. Gansner, "Bunch: a clustering tool for the recovery and maintenance of software system structures," in *Proceedings IEEE International Conference on Software Maintenance - 1999 (ICSM'99). 'Software Maintenance for Business Change' (Cat. No.99CB36360)*, pp.50–59, Aug 1999.