به نام خداوند بخشنده‌ی مهربان

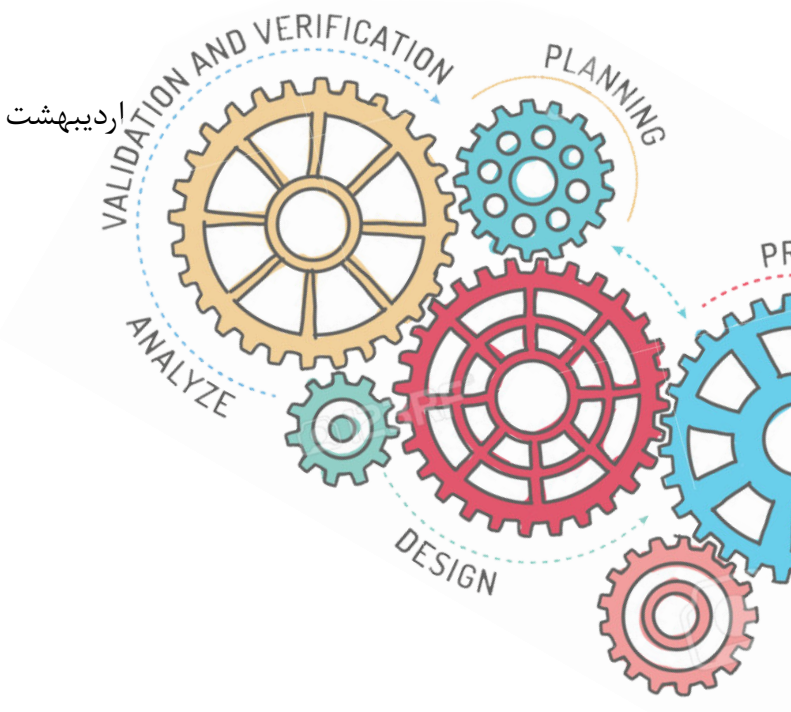
مهندسی نرم‌افزار

تمرین سری سه

استاد درس: دکتر حبیبی

محمّدمهدی فاریابی
۹۳۱۰۱۹۵۱

اردیبهشت ماه ۱۳۹۷



۱ پرسش اول

موارد خواسته‌شده در مورد مهندسی نیازمندی امنیتی را شرح می‌دهیم:

۱.۱ مدل گراف حمله(Attack Graph)

دکتر Cynthia Phillips به همراه یکی از همکارانشان در سال ۱۹۹۸ مدلی گرافی را برای بررسی نقاط ضعف در شبکه‌های کامپیوتری معرفی کردند. این مدل مکانیزمی برای بررسی حملات و نسبت دادن آنها به ماشین‌های داخل شبکه و مهاجمین ارائه می‌کند.

این مدل بر اساس طرح‌ها و قدمات حملات شناخته‌شده‌ی قبلی در سیستم‌های دیگر، پیکربندی شبکه‌ی سیستم فعلی و بررسی و زیر نظر گرفتن شبکه توسعه می‌یابد. [۲]

گراف حمله بخش مهمی از تحلیل نفوذ پذیری در سیستم‌های تحت شبکه است و از آنجایی که تهیه‌ی چنین مدلی به صورت دستی آسان نیست، روشی اتوماتیک وغیر رسمی برای ایجاد آن در طراحی‌های iterative پیشنهاد شده است. در این روش، اهداف سیستم، اهداف متهاجمین احتمالی و مسیرهای تهاجم با احتمال موفقیت بالا را مدل سازی کرده و مورد بررسی و مستندسازی قرار می‌دهد. [۳]

۲.۱ UMLsec

UMLsec (با secureUML اشتباه گرفته نشود) افزونه‌ای به زبان مدلسازی UML است که از طریق افزودن مجموعه‌ای از stereotype ها و tag ها و constraint ها به زبان UML سعی در گنجاندن مفاهیمی و اطلاعات مربوط امنیتی در دل نمودارهای UML مانند State Diagram ، Sequence Diagram و Activity Diagram دارد. هدف اصلی این افزونه کپسوله کردن مفاهیم امنیتی و رخدادهای پرتکرار در حوزه‌ی امنیت و ارائه‌ی آن به توسعه‌دهندگان در قالب مجموعه‌ای منسجم است.

به وسیله‌ی طراحی غنی شده با این افزونه می‌توان تهدیدهای امنیتی خاص را مدل کرده، بررسی کرد و تاثیر آنها را بر امنیت و کارایی کل سیستم در زمان اجرای این تحریک‌ها مورد مطالعه قرار داد. [۴]

۳.۱ مورد سوء استفاده(Abuse Case)

مدل سوء استفاده مدلی است که از نوتیشن UML با نمادهای برعکس شده استفاده می‌کند. هدف این کار نشان دادن functionality است که از سیستم انتظار **نمی‌رود.**

این مدل بر پایه‌ی زبان مدل‌سازی UML بنا شده است.

Use Case ها موارد کاربرد مورد انتظار و مثبت سیستم هستند و کارایی سیتم نرم‌افزاری را نشان می‌دهند. در مقابل Abuse Case ها جزئیات تعاملاتی با سیستم را نشان می‌دهند که برای سیستم، محیط آن یا کاربران آن مخاطراتی را به همراه خواهند داشت.

یک Abuse Case مراحلی را نشان می‌دهد که از امکانات مجاز یک سیستم برای اتمام سوء استفاده طی می‌شود. [۴]

۲ پرسش دوم

برای هر کدام از حوزه‌های خواسته شده دو الگوی طراحی را شرح می‌دهیم:

۱.۲ پردازش ابری

۱.۱.۲ اتوماسیون مدیریتی

- **نام**
اتوماسیون مدیریتی یا Automated Administration

- **توضیحات**
چگونه باید وظایف مدیریتی رایج را به صورت مداوم و اتوماتیک در پاسخ به رخدادهای از پیش تعیین شده به اجرا در آورد.

- **مشکل**
منابع در سیستم‌های فناوری اطلاعات درستخوش تعداد زیادی فعالیت مدیریتی تکراری و مداوم می‌شوند. سپردن این فعالیت‌ها به عامل انسانی احتمال بروز خطا و مشکل در عملیات را افزایش می‌دهد.

- **راه‌حل**
مراحل کار فعالیت‌های مدیریتی مناسب خودکارسازی است. برای این منظور script هایی ایجاد شده و در یک سیستم خودکار اجرا کننده قرار داده می‌شوند.

- **کاربرد**
یک موتور اتوماسیون هوشمند در کنار سیستم اطلاعاتی قرار می‌گیرد تا فعالیت‌های تکراری و مداوم مدیریتی را متناسب با شرایط اجرا کند.

- **منبع**
`http://cloudpatterns.org/design_patterns/automated_administration`

۲.۱.۲ پرداخت به میزان مصرف

- **نام**
پرداخت به میزان مصرف Pay as You Go

- **توضیحات**
چگونه باید از یک مشتری، متناسب با هزینه‌ی مصرف منابع او هزینه دریافت کرد.

- **مشکل**
خریداری یا اجاره‌ی یک منبع IT ممکن است با هزینه‌هایی همراه باشد که با میزان مصرف و استفاده‌ی واقعی از آن منبع متفاوت باشد.

- **راه‌حل**
سیستمی معرفی و استفاده می‌شود که میزان مصرف را اندازه‌گیری کرده و متناسب با آن تخصیص هزینه‌ی لازم را به کاربر انجام دهد.

- **کاربرد**
سیستم نظارت در زمان اجرا و مصرف و سیستم محاسبه‌ی هزینه‌ها به صورت آنی مصرف کاربر را زیر نظر گرفته و هزینه‌های لازم را صادر می‌کنند.

- **منبع**
`http://cloudpatterns.org/design_patterns/pay_as_you_go`

۲.۲ تحمل اشکال(Fault Tolerant)

۱.۲.۲ من زنده‌ام

- **نام**
من زنده‌ام یا I am alive

- **توضیحات**
سیستمی داریم که در آن هزینه‌ی ارتباط با واحد نظارت قابل تحمل و پهنای باند مورد نیاز برای ارتباط با سیستم نظارت به وضوح کمتر از حداکثر پهنای باند است. در این سیستم زمان یک تعامل با واحد نظارت محدود یا شناخته شده است.

- **مشکل**
ممکن است یک سرویس یا سیستم حین کار خاموش شده یا از دور خارج شود. نیاز به وجود مکانیزمی برای تشخیص سریع و اقدام برای این مورد حس می‌شود.

- **راه‌حل**
سیستم نظارت به صورت دوره‌ای و منظم سیستم مورد نظارت را چک کرده و از صحت عملکرد آن مطلع می‌شود. به این ترتیب این سیستم اطلاعات خود را از الگوهای خطایی سیستم مورد نظارت به تدریج کامل می‌کند. در صورت بروز خطا، سیستم پشتیبانی به سرعت درگیر و فعال می‌شود.

- **کاربرد**
سیستم چک کردن خودکار زنده‌بودن در کنار و به عنوان عضوی از سیستم نظارت فعال شده و به کار می‌پردازد.

- **منبع**
https://pdfs.semanticscholar.org/117c/c559abe000542128da6f24b0e31319db1eca.pdf

۲.۲.۲ افزونگی

- **نام**
افزونگی یا Redundancy

- **توضیحات**
ممکن است در یک سیستم نرم‌افزاری بنا به دلایلی، سیستم از کار بیفتد یا اطلاعات از بین برود.

- **مشکل**
از دست رفتن اطلاعات یا سایر منابع به علت بروز خطا ممکن است در یک سیتم نرم‌افزاری پیش بیاید.

- **راه‌حل**
با نگهداری نسخه‌های مختلفی از یک منبع مانند پایگاه داده یا منابع فیزیکی و سخت افزاری و یا یم سرویس نرم‌افزاری می‌توان در صورت از کار افتادن نسخه‌ی قبلی، نسخه‌ی دیگری را جایگزین آن کرد.

- **کاربرد**
سیستمی برای نظارت و مدیریت نسخ مختلف یک منبع و همچنین تقسیم کار بین آنها ایجاد شده و مدیریت این منابع را بر عهده می‌گیرد.

- **منبع**
https://pdfs.semanticscholar.org/9341/0c000b781ba73210aa0d2958195a6193e64b.pdf

۳.۲ تست مکانیزه(Automated Testing)

۱.۳.۲ صفحه شیء

- **نام**
صفحه شی یا Page-Object

- **توضیحات**
برای تست عملکرد در رابط گرافیکی لازم است اطلاعات مختلفی مانند تعاملات با صفحه و آنچه در صفحه مشاهده می‌شود دخیل شود.

- **مشکل**
تست رابط گرافیکی و عملکرد سیستم در آن ممکن است به دلیل ماهیت تعاملی آن با انسان بسیار پیچیده و سخت باشد

- **راه‌حل**
با نگهداری تمامی اطلاعات تست مانند تعاملات با صفحه و وردوی‌ها و تغییرات رابط گرافیکی در یک شی واحد می‌توان بررسی تغییرات و انجام تست را بسیار ساده تر کرد.

- **کاربرد**
سیستمی برای تقسیم بندی جزئی تمامی بخش‌های تعامل کاربر با محصول مانند صفحه‌ی نمایش و موقعیت اشاره گر ایجاد شده که اطلاعات را در قالب یک سری frame و cell ذخیره می‌کند. این اطلاعات برای انجام تست و تهیه‌ی مدل‌های تست مورد استفاده قرار می‌گیرد.

- **منبع**
`https://www.automatetheplanet.com/page-object-pattern/`

۲.۳.۲ صورت

- **نام**
صورت یا Facet

- **توضیحات**
برای تست عملکرد یک سیستم گاهی لازم است تستی مرکب شامل تست روی چندین صفحه انجام شود.

- **مشکل**
تست یک عملکرد سیستم نرم‌افزاری ممکن است به علت ماهیت مرکب و طولانی آن شامل چندین صفحه‌ی کاربری باشد و باعث دشواری آزمون شود.

- **راه‌حل**
با نگهداری اطلاعات چندین تست در یک شی facet می‌توان تست را روی این شی انجام داد. به این ترتیب هر facet شامل اطلاعات چند page-object خواهد بود و مکانیزمی برای اجرای یک باره‌ی کل تست ارائه خواهد شد.

- **کاربرد**
اطلاعات مرکب تست توسط سیستم تست به صورت خودکار ضبط و بسته بندی خواهد شد و اجرای تست مرکب در دفعات آتی همانند اجرای تست ساده خواهد بود.

- **منبع**
`https://www.automatetheplanet.com/facade-design-pattern/`

۳ پرسش سوم

روش‌های خواسته‌شده برای مهندسی نیازمندی‌ها در مهندسی نرم‌افزار را شرح می‌دهیم:

۱.۳ مهندسی نیازمندی هدف محور(Goal Oriented)

در مهندسی نیازمندی‌های هدف محور اینکه سیستم برای چه مورد نیاز است و هدف اصلی آن چیست، مبنا و پایه‌ی اصلی استخراج و مهندسی نیازمندی‌هاست.

بسیاری از متودولوژی‌های نرم‌افزاری از زمان‌های گذشته تا به امروز از این روش و معیار برای مهندسی نیازمندی ها در پروژه استفاده می‌کنند.

در این روش ابتدا اهداف از سیستمی که قرار است تولید شود بیان می‌شوند. سپس سیستم فعلی در ابعاد مختلف سازمانی، اجرایی مورد بررسی دقیق قرار می‌گیرد و میزان توجه سیستم به نیازمندی‌های قبلی مورد بررسی قرار می‌گیرد. در نهایت سیستم جدید با هدف پاسخگویی بهینه به نیازهای کاربران مورد توسعه قرار می‌گیرد.

در چنین روشی خیلی اوقات نیاز به مدلسازی سطح بالاتری نیاز است. به این ترتیب این مدلسازی سطح پایین تر بسیاری از کلیات مانند تعاملات و نیازمندی‌های آنها را به وضوح توضیح نمی‌دهد.

https://ieeexplore.ieee.org/abstract/document/948567/

۲.۳ مهندسی نیازمندی عامل محور(Agent Oriented)

به تازگی روش جدیدی برای مهندسی نیازمندی‌ها مطرح شده که در آن به جای مطرح کردن اهداف سیستم به عنوان مرجع مدلسازی،عامل‌های سیستم، تعاملات بین آنها و سیستم ساخته شده از ارتباطشان مرجع قرار می‌گیرد.

عامل‌ها در چنین سیستمی با محیط، با یکدیگر و با عوامل انسانی تعامل می‌کنند، هوشمندی دارند، به دنبال موقعیت انجام وظایف خود هستند و حتی می‌توان به صورت انتزاعی به آنها روحیات و سطح کارایی نسبت داد. به این صورت می‌توان استخراج نیازمندی‌ها را بسیار سطح بالا و به زبانی بسیار نزدیک تر به زبان انسان انجام داد. چنین کاری انعطاف مدل‌های ایجاد شده و درک آنها را بسیار ساده تر می‌کند و بسیار قدرتمند تر از مهندسی نیازمندی‌های هدف محور است.

https://pdfs.semanticscholar.org/0dcd/d1324db2a6888010332bd747520611f28ab0.pdf

۴ پرسش چهارم

سطوح مختلف جفت شدگی [۱] و پیوستگی [۲] را به همراه مثال تشریح می‌کنیم:

۱.۴ سطوح مخالفت جفت شدگی

۱. **بالاترین سطح جفت‌شدگی**

(آ) **جفت‌شدگی در محتوا**[۳]

ماژول A به محتوا و بخش‌های private ماژول B دسترسی پیدا می‌کند. این کار ممکن است از طریق اجرای یک دستور GOTO در ماژول A اتفاق بیفتد. این سطح از جفت شدگی به هیچ عنوان نباید اجازه داده شود

۲. **جفت‌شدگی بالا**

(آ) **جفت‌شدگی عادی**[۴]

دو ماژول یا بیشتر که به داده‌ی Global مشترکی دسترسی دارند، با یکدگیر جفت شدگی عادی دارند. این جفت شدگی مورد تایید نیست ولی شاید اجتناب ناپذیر باشد.

(ب) **جفت‌شدگی خارجی**[۵]

چند ماژول که به یک دستگاه IO دسترسی دارند، با یکدیگر جفت شدگی خارجی دارند. این جفت شدگی مورد تایید نیست ولی شاید اجتناب ناپذیر باشد.

۳. **جفت‌شدگی میانه**

(آ) **جفت‌شدگی در کنترل**[۶]

اگر ماژول A با پاس دادن اطلاعاتی به ماژول B آن را کنترل کند گوییم ماژول‌های A و B با یکدیگر جفت شدگی در کنترل دارند. مثلا خروجی یک رویه در ماژول A وضعیت را در یک حلقه در ماژول B تعیین کند.

این جفت شدگی قابل قبول است ولی مدل‌های طراحی و پیاده سازی باید به صراحت وجود این کنترل را تایید کنند.

۴. **جفت‌شدگی کم**

(آ) **جفت‌شدگی در داده**[۷]

گوییم دو ماژول جفت شدگی در داده دارند اگر تنها تعاملات این دو در پاس دادن پارامترهایی ساده به رویه‌های یکدیگر باشد. این جفت شدگی تنها جفت شدگی عادی و قابل قبول برای دو ماژول است و جفت‌شدگی‌های دیگر تنها در صورتی که نیاز باشد مورد اسفتاده هستند.

(ب) **جفت‌شدگی در داده‌ی پیچیده**[۸]

گوییم دو ماژول جفت شدگی در داده دارند اگر تنها تعاملات این دو در پاس دادن پارامترهایی مرکب به رویه‌های یکدیگر باشد.

۵.

---

[1] Coupling
[2] Coheision
[3] Content Coupling
[4] Common Coupling
[5] External Coupling
[6] Control Coupling
[7] Data Coupling
[8] Stamp Coupling

۶. **پایین ترین سطح جفت شدگی**

گوییم دو ماژول پایین ترین سطح جفت شدگی را دارند اگر این دو اصلا هیچ تعاملی با یکدیگر نداشته باشند.

http://pages.cpsc.ucalgary.ca/~eberly/Courses/CPSC333/Lectures/Design/coupling.html

۲.۴ سطوح مختلف پیوستگی

۱. **پیوستگی کم**

(آ) **پیوستگی اتفاقی**[9]

ماژولی که کارهایی را انجام می‌دهد که هیچ ربط واضحی به یکدیگر ندارد پیوستگی اتفاقی دارد. مانند ماژولی که دو وظیفه‌ی تعمیر سیستم و آپلود داده را بر عهده دارد. چنین ماژول‌هایی باید حذف و با چندین ماژول خاص منظور جایگزین شوند.

(ب) **پیوستگی منطقی**[10]

ماژولی پیوستگی منطقی دارد که رویه‌هایی در آن در یک دسته‌ی عمومی قرار داشته باشند و انتخاب جزئی از بین این دسته‌ها از بیرون ماژول با فراخوانی یکی از آنها انجام شود. مثلا ماژولی که یک رویه برای نوشتن در حافظه‌ی جانبی و یک رویه برای نوشتن در دیسک دارد. معمول نیست یک کاربر هر دوی این دستورات را در یک کار خود انجام دهد. برای درست کردن این نکته‌ی منفی می‌توان یک رویه در اختیار بیرون قرار داد که با گرفتن یک پارامتر جزئیات عمل را متوجه می‌شود و رویه‌های قبلی را private کرد.

(ج) **پیوستگی زمانی**[11]

ماژولی پیوستگی زمانی دارد که دارای رویه‌هایی باشد که همگی در زمان به یکدیگر مرتبط اند. مثلا ماژولی را در نظر بگیرید که برای خاموش کردن سیستم استفاده می‌شود و رویه‌ی بستن فایل‌های باز و همچنین قطع اتصال شبکه را ارائه می‌دهد. حین خاموش شدن سیستم این دو کار باید در توالی انجام شوند و دادن انتخاب نسبت به انجام آنها به کاربر کار درستی نیست. برای حل مشکل رویه‌ی خاموش شدن سیستم ارائه می‌شود که این دو رویه را مورد استفاده قرار خواهد داد. این دو نیز private خواهند شد.

۲. **پیوستگی میانه**

(آ) **پیوستگی در فرایند**[12]

چنین ماژولی دارای رویه‌هایی متفاوت و بعضا نامرتبط است که کنترل در آن از یکی از رویه‌ها به بعدی منتقل می‌شود. مانند آماده سازی پوشه‌های مورد نیاز و بعد از آن تولید نسخه‌ی پشتیبان.

این پیوستگی مورد قبول است و همچنین بهتر از پیوستگی زمانی است. چرا که روند منطقی بین کارها قابل مشاهده است. ولی کماکان دلیل منطقی برای قرار دادن این کارها در یک ماژول وجود ندارد.

(ب) **پیوستگی در ارتباط**[13]

ماژولی در سطح پیوستگی در ارتباط است که رویه‌هایی داشته باشد که همگی به یک داده‌ی ورودی دسترسی دارند یا همگی یک بخش از ساختمان داده را مورد استفاده قرار می‌دهند. مانند رویه‌های پیدا کردن نام کتاب و پیدا کردن تعداد صفحات کتاب.

یا ماژولی که تمام رابط یک داده ساختار را ارائه کند.

(ج) **پیوستگی در توالی**[14]

ماژولی که دارای رویه‌هایی باشد که خروجی یک رویه به عنوان ورودی رویه‌ی بعدی مورد استفاده قرار بگیرد.

مانند شستن لباس‌ها، خشک کردن لباس‌های شسته شده و در نهایت اتو کردن لباس‌های خشک شده.

---

[9] Coincidental Cohesion

[10] Logical Cohesion

[11] Temporal Cohesion

[12] Procedural Cohesion

[13] Communicational Cohesion

[14] Sequential Cohesion

۳. پیوستگی بالا

(آ) **پیوستگی در کاربرد**[15]

ماژولی در کاربرد پیوستگی دارد که تنها رویه‌هایی را شامل شود که برای انجام تنها یک وظیفه‌مندی مربوط به مساله طراحی و ایجاد شده اند.

مثلا ماژولی که سینتکس یک ورودی را بررسی و تایید می‌کند یا ماژولی که برای تعیین صندلی در سینما مورد استفاده قرار می‌گیرد

http://pages.cpsc.ucalgary.ca/~eberly/Courses/CPSC333/Lectures/Design/cohesion.html

[15] Functional Cohesion

مراجع

[1] "**Why is Feature Driven Development considered an Agile methodology?**", Software Engineering - StackExchange, retrieved 20 farvardin 1397

`https://softwareengineering.stackexchange.com/questions/199021/why-is-feature-driven-development-considered-an-agile-methodology`

[2] C. Phillips and L. P. Swiler, "**A graph-based system for network-vulnerability analysis**, In Proc. of NSPW'98, pages 71–79. ACM, 1998.

[3] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing, "**Automated generation and analysis of attack graphs**, In SP '02: Proceedings of the 2002 IEEE Symposium on Security and Privacy, page 273, Washington, DC, USA, 2002. IEEE Computer Society.

[4] Golnaz Elahi, "**Security Requirements Engineering: State of the Art and Practice and Challenges**", university of torrento, retrieved 15 ordibehesht 1397

`http://www.cs.toronto.edu/~gelahi/DepthPaper.pdf`