International Conference on
Engineering & Applied Sciences
UAE - DUBAI
10 March 2016
ICEASCONF
کنفرانس بین المللی
مهندسی و علوم کاربردی
امارات - دبی
۲۰ اسفند ۹۴

محاسبه قابلیت اطمینانِ مولفه‌های ساختاری الگوی طراحی MVC

با استفاده از شبکه پتری رنگی

محمدجواد نجفی‌سولاری[۱]

۱. دانشگاه آزاد اسلامی، واحد بوشهر ،کارشناس ارشد نرم افزار، اهواز، ایران.

najafisolari.javad@gmail.com

چکیده

امروزه یکی از چالش‌های مهندسین نرم‌افزار، توسعه و ارتقاء محصولات نرم‌افزاری است، بطوری که بتوانند نیازهای سیستم را برآورده و با اندک هزینه و زمان ، سیستمی مطمئنی در اختیار مشتریان قرار دهند. یکی از مهم‌ترین صفات کیفیتی یک سیستم نرم‌افزاری، قابلیت‌اطمینان است، که با ارزیابی آن در مراحل اولیه ی توسعه، معمار قادر است مولفه‌های بحرانی را پیش‌بینی کند. در این مقاله ساختار الگوی معماری MVC ، که یکی از الگوهای پرکاربرد و شناخته شده است را به کمک کارت‌های CRC و نمودار مولفه از زبان مدلسازی یکنواخت، توصیف کرده و پارامترهای قابلیت اطمینان به صورت کلیشه و برچسب به آن حاشیه‌نویسی میشود. سپس با استفاده از شبکه پتری رنگی، مدل واقعی به مدل اجرایی تبدیل شده و قابلیت‌اطمینان برای هر یک از مولفه‌های این الگو محاسبه می‌گردد.

واژگان کلیدی: الگوی معماری MVC ، قابلیت اطمینان ، شبکه پتری رنگی، نمودار مولفه، کارت CRC.

۱. مقدمه

معماری نرم‌افزار به عنوان اولین محصول نقش مهم و مستقیمی در توسعه سیستم‌های نرم‌افزاری پیچیده ایفا می‌کند. با افزایش و پیشرفت محیط‌های کاربردی جدید، ارزیابی نیازهای وظیفه مند و غیروظیفه‌مند به منظور رسیدن به نیازهای مشتریان ضروری است. تحلیل و ارزیابی نیازهای غیروظیفه‌مند فعالیتی است که بایستی امروزه در فرایند توسعه سیستم‌های نرم‌افزاری بزرگ و پیچیده مورد نظر قرار گیرد[۱]. در واقع با ارزیابی می توان اطمینان پیدا کرد که تا چه حد معماری میتواند اهداف کیفیتی فوق را برآورده نماید.

انتخاب الگو و در نهایت ارزیابی آن تصمیمات مهمی هستند که محققین معماری نرم‌افزار بر روی آن متمرکز شده‌اند. اگر معماری به درستی انتخاب نشود یک سیستم با کیفیت پایین تولید می‌شود که در نتیجه باعث به هدر رفتن منابع پروژه ار قبیل زمان و هزینه می‌گردد .

یکی از الگوهای معماری نرم‌افزار MVC می‌باشد. که برای سیستم‌های تعاملی مورد استفاده قرار می‌گیرد. این الگو یک برنامه تعاملی را به سه مولفه تقسیم می‌کند و معمولا به عنوان یکی از مدل‌های توسعه، در طراحی سیستم‌های توزیعی بسیار استفاده می‌شود[۲].(ژانگ و شن، ۲۰۱۲).

با ایجاد یک مدل قابل اجرا از این الگو می توان قبل از ورود به فاز پیاده سازی نیازهای کیفیتی را ارزیابی نمود. قابلیت اطمینان به عنوان یکی از نیاز های غیروظیفه‌مند ، باعث تغییرات قابل توجهی در هر مرحله از چرخه زندگی نرم افزار به خصوص در مراحل اولیه ی توسعه آن میشود، در این راستا در این مقاله سعی شده تا این صفت در مراحل اولیه ی توسعه مورد ارزیابی قرار گیرد[۳].

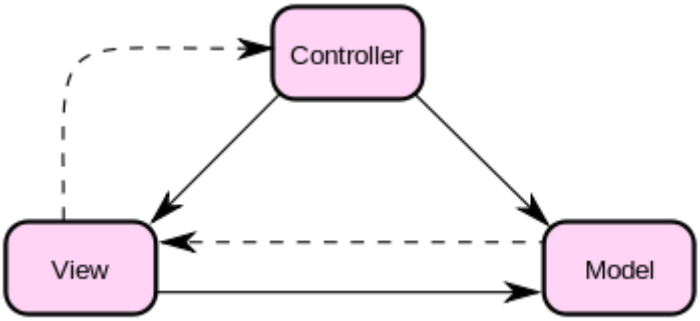
در ادامه‌ی مقاله در بخش دوم مروری جزئی بر الگوی معماری MVC، شبکه پتری رنگی به عنوان یک مدل قابل اجرا، قابلیت اطمینان به عنوان یک نیاز غیروظیفه‌مندی به عنوان بستر کار خواهد داشت و سپس کارهای انجام شده در این راستا در بخش سوم مورد بررسی و روش پیشنهادی نیز در بخش چهارم شرح داده خواهد شد. و نهایتا در بخش پنجم و ششم به مطالعه موردی و نتیجه گیری می پردازد.

۲. بستر کار:

زمینه انجام کار شامل الگوی طراحی MVC ، شبکه پتری رنگی به عنوان مدل قابل اجرا و صفت کیفیتی قابلیت اطمینان است که به اختصار به بررسی هر یک پرداخته شده است.

۲-۱. الگوی معماری MVC

الگوهای معماری راهی برای رسیدن به طراحی با ساختاری خوب هستند. و مزیت مهم آنها قابلیت استفاده مجدد میباشد. الگوی معماری MVC یک برنامه تعاملی را به سه مولفه تقسیم می‌کند. "مدل"، که شامل عملکردهای اصلی و داده‌هاست. "دید"، اطلاعات را برای کاربر به نمایش می‌گذارد. "کنترلر"، ورودی‌های کاربر را رسیدگی می‌کند. که دید و کنترلر با هم واسط کاربر را تشکیل می‌دهند[۴]. این معماری پیاده‌سازی سطوح داده و سطوح ابزار را بسیار خوب از هم تفکیک می‌کند و معمولا به عنوان یکی از مدل‌های توسعه در سیستم‌های توزیعی استفاده می‌شود[۵]. شکل ۱ شمای کلی از این معماری را ارائه میدهد.



شکل ۱: نمایش الگوی کلی معماری MVC

۲-۲. شبکه پتری رنگی به عنوان مدل قابل اجرا

نظریه شبکه پتری برای اولین بار توسط کارل آدام پتری در سال ۱۹۶۲ ارائه گردید. این شبکه ها به عنوان ابزاری برای مدلسازی سیستم‌ها استفاده می‌گردند. در میان مدل‌های رسمی ، شبکه‌های پتری ابزار قدرتمندی جهت مدلسازی همزمانی هستند. این شبکه ها یک زبان گرافیکی برای مدل‌های ساختاری از سیستم های همروند و آنالیز ویژگی ها ارائه میدهند. از دیگر خصوصیات شبکه های پتری قابل اجرا بودن آن است. که از همین خاصیت می توان برای ارزیابی رفتار و کارایی یک سیستم استفاده نمود[۶]. شبکه پتری دارای بسط های متعددی می باشد که می توان شبکه پتری رنگی، شبکه پتری رنگی زمانی، شبکه پتری تعمیم یافته ، شبکه پتری تصادفی و ... را نام برد.

۲-۳. صفت کیفیتی قابلیت اطمینان

قابلیت اطمینان یکی از مهمترین صفات کیفیتی یک سیستم نرم‌افزاری است. که پایین بودن آن منجر به ارائه سرویس های نا مطلوب و حتی غیر قابل اطمینان می شود. تعریفی که از قابلیت اطمینان در IEEE بیان شده: احتمال درست کار کردن نرم افزار، عاری از هر گونه خطا در یک زمان مشخص و تحت شرایط خاص است. ارزیابی این صفت در سطح معماری نرم افزار باعث کاهش هزینه خطا می‌شود[۷].

۳. کارهای مرتبط

[۸] به منظور ارزیابی کمی قابلیت اطمینان نرم‌افزار ، یک مدل قابلیت اطمینان مولفه، مبتنی بر شبکه پتری تصادفی ارائه داده‌اند و زنجیره مارکوف را متناظر با شبکه پتری تصادفی بدست آورده‌اند. سپس مقدار قابلیت اطمینان طبق توزیع احتمال تجمعی زنجیره مارکوف بدست می‌آید. [۹] از ترکیب سبک‌های معماری لایه‌ای و لوله‌وفیلتر برای موازنه صفات متعامد استفاده کرده است. در این مقاله صفت کیفی قابلیت اطمینان مورد ارزیابی قرار گرفته است و برای اینکه محاسبه و پاسخ عددی از این صفت بدست آید یک مدل اجرایی از معماری با شبکه پتری رنگی ارائه شده است و سپس این مدل به زنجیره مارکوف نگاشت داده شده و با کمی‌سازی در زنجیره مارکوف صفت قابلیت اطمینان به صورت عددی محاسبه گردیده است. [۱۰](بهزاد سلیمانی نیسیانی و همکاران، ۱۳۹۰) در مقاله خود مدل‌های قابلیت اطمینان را طبق چرخه حیات توسعه نرم افزار رده‌بندی نموده‌اند و تعدادی از معیارها و اولویت‌ها برای انتخاب مدل قابلیت اطمینان را تعریف و همچنین الگوریتمی برای استفاده از معیارها بیان کرده‌اند.[۱۱] همچنین بیان شده است که اگر قابلیت اطمینان مولفه ها در دسترس باشد سناریو ها می توانند برای ارزیابی قابلیت اطمینان اولیه سیستم مورد استفاده قرار گیرند. و به همین خاطر

روشی خودکار برای پیشی بینی قابلیت اطمینان سیستم نرم افزاری ارائه شده است. این رویکرد در برگیرنده‌ی بسط دادن احتمال شکست مولفه ها و احتمال گذار سناریو استخراج شده از مشخصات عملیاتی سیستم به یک مدل است. با بسط خصوصیات سناریو، مدل رفتار احتمالی برای هر مولفه ساخته شده و سپس به صورت موازی یک مدل برای سیستم تشکیل می شود و در نهایت یک مدل قابلیت اطمینان کاربرد محور برای محاسبه پیش بینی قابلیت اطمینان مدل رفتاری سیستم شرح داده شده است. [۱۲] یک روش ارزیابی قابلیت اطمینان با استفاده از سیستم‌های مبتنی بر فازی - عصبی و اثر بخش در ارزیابی قابلیت اطمینان ارائه داده‌اند. همچنین مقایسه ای بین شبکه عصبی مبتنی بر رشد قابلیت اطمینان نرم افزار و منطق فازی مبتنی بر رشد قابلیت اطمینان نرم افزار بر اساس یک فرایند پواسون همگن برای ارزیابی قابلیت اطمینان نرم افزار انجام شده است. [۱۳] از یک مدل مبتنی بر RVM برای پیشی بینی قابلیت اطمینان نرم افزار مورد نظر برای گرفتن ارتباط داخلی بین داده زمان شکست نرم افزار و داده زمان نزدیک شکست استفاده شده است. ارزیابی و مقایسه اثر بخشی RVM در زمان شکست پیش بینی شده برای محصولات نرم افزاری انجام شده و از روش آزمون Mann-Kendall برای دقت پیش بینی استفاده شده است. [۱۴] یک روش برای نشان داده سیستم نرم افزاری شی گرا با استفاده از یک شبکه پیچیده وزن دار به منظور جذب ویژگی های ساختاری با توجه به قابلیت اطمینان پیشنهاد شده است. لبه ها و گره ها با توجه به پیچیدگی کلاس و وابستگی های خود ، مدل شده اند. همچنین معیارهای تبدیل نظریه گراف بر روی شبکه با هدف ارزیابی سیستم نرم افزاری اعمال می‌شود. این روش با ارزیابی سیستم قادر به شناسایی اجزای نرم افزاری که اصول طراحی را نقض می کنند می باشد.

۴. روش پیشنهادی

۴-۱. نگاشت الگوی طراحی MVC به کارت CRC

تا کنون در مورد بخش‌های MVC فقط در شرایط عمومی صحبت کردیم. در این قسمت ما در مورد نقش‌ها و مسئولیت‌های سه بخش MVC توضیح خواهیم داد که هر یک به عنوان یک مولفه دیده می‌شود[۴].

مولفه ی "مدل"، هسته اصلی برنامه شامل داده‌های آن و توابع مربوط به داده‌هاست. در واقع روش‌های دسترسی به داده را ارائه می‌دهد. این روش‌ها از طریق کنترل در پاسخ به درخواست کاربر فراخوانی می‌شوند. مولفه‌ی"مدل" همچنین شامل توابعی است که برای دسترسی به داده‌ها، دیدها از آنها استفاده و اطلاعات را به کاربران نمایش می‌دهند. هر مولفه ی "دید" یک مولفه ی "کنترل" مناسب ایجاد می‌کند. به طوری که یک رابطه یک به یک بین مولفه ی "دید و کنترل" به وجود می آید. مولفه ی "دید" اغلب توابعی را ارائه می‌دهد که اجازه دهد. مولفه ی "کنترل" صفحه نمایش را دستکاری کند. این عملیات باعث می‌شود که کاربر روی "مدل" تاثیر نگذارد. همچنین مولفه ی "کنترل" ورودی کاربر را به عنوان رخداد می‌پذیرد[۴]. در این راستا به منظور ارائه مولفه ها و ارتباطات آنها از کارت های CRC استفاده شده است کارت‌های CRC می‌توانند مسئولیت های یک کلاس و چگونگی ارتباط آنها با دیگر کلاس ها را به طور ایستا نشان دهند[۵]. که این همان مولفه‌ها هستند. مطابق تقسیم‌بندی مولفه‌های معماری این الگو می‌توان هر کارت CRC را به هر مولفه نسبت داد.

۴-۲- تبدیل کارت‌های CRC به نمودار مولفه

کارت‌های CRC یک نمای ساختاری از معماری را نشان می‌دهد اما قابلیت اضافه شدن پارامترهای اضافی را ندارد از آنجایی که نمودار مولفه به عنوان یکی از نمودارهای زبان مدلسازی یکنواخت ، قابلیت اضافه شدن پارامترهای نیاز های غیروظیفه‌مند را دارا می‌باشد لذا در این مقاله ما کارت CRC را به نمودار مولفه تبدیل می‌کنیم. برای انجام این کار مراحل زیر را که در مقاله [۱۵ و ۱۶] مورد استفاده قرار گرفته ، دنبال می‌کنیم:

- گام اول: هر کارت CRC به یک مولفه تبدیل می‌شود.
- گام دوم: در هر کارت CRC در مقابل هر مسئولیت، مولفه همکار آن بیان شده است. که این دو مجموعه با هم نشان‌دهنده ارتباط بین مولفه‌هاست.

برای هر مسئولیتی که مقابل آن ، نام مولفه ذکر شده باشد یعنی ارتباط "نیاز دارد" و با نام مسئولیت حاشیه‌نویسی می‌شود. و دیگر مولفه ها که در مقابل آنها مسئولیتی قرار ندارد و در واقع فراهم‌کننده نیاز مولفه‌های دیگر هستند هر یک با نام نیاز را فراهم می‌کند حاشیه‌نویسی می‌شوند.

۴-۳- مدل رسمی از ساختار الگوی طراحیMVC و حاشیه نویسی پارامتر قابلیت اطمینان

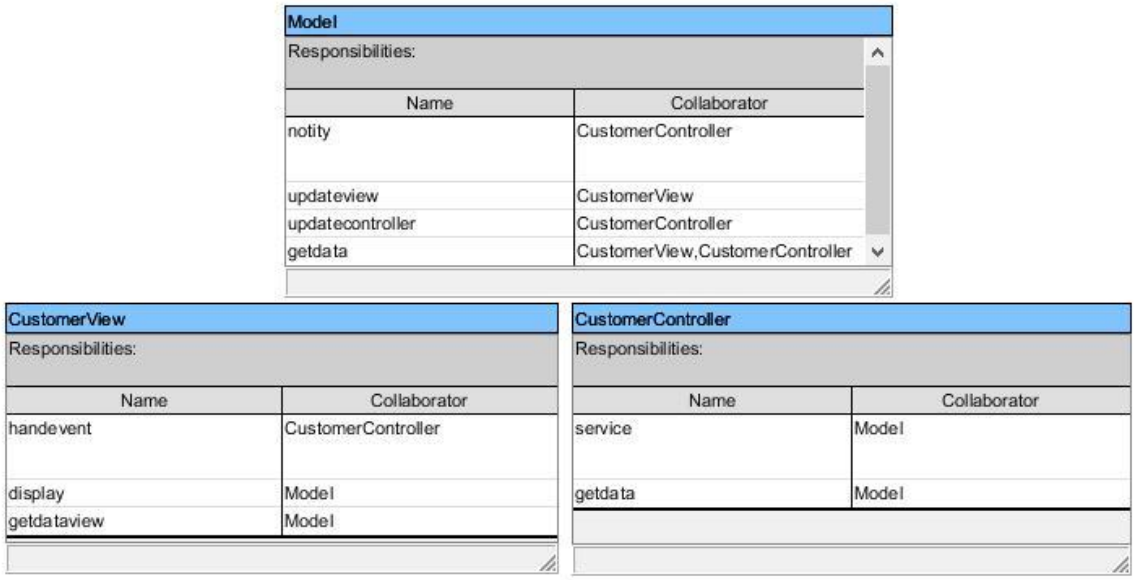
برای تبدیل نمودار مولفه، به شبکه پتری از روش استفاده شده در [۱۶] (شمس، ۱۹۹۶) استفاده می کنیم. در این روش ابتدا باید ترتیب وظایف هر مولفه با استفاده از نقش‌ها و عبارات مسیر ، پالایش گردد. در واقع این عبارات مسیر هستند که ترتیب دقیق میان عملیات مولفه‌ها را مشخص می‌کنند. سپس هر مولفه به صورت جدا گانه به شبکه پتری تبدیل شده و نهایتا بر اساس نوع ارتباط مولفه ها با یکدیگر، شبکه های پتری حاصله با هم ترکیب می‌شوند. سپس در ادامه پارامتر های قابلیت اطمینان به هر مولفه کلیشه <<REcomponent>> با برچسب REcomfailprop حاشیه نویسی شده که در واقع با این کار به هر مولفه احتمال شکست را نسبت می دهیم. همچنین می توان به ارتباط مولفه ها با یکدیگر ، کلیشه <<REconnctor>> با برچسب REconnfailprop نیز الحاق کرد که با این کار احتمال شکست بین مولفه ها و تعداد تعاملات بیان می شود.

۵. مطالعه موردی

مطالعه موردی اختصاص به دسترسی ، مشتریان به سیستم نظرسنجی دارد. که در اینجا بخشی از زیر سیستم پیاده سازی گردیده است. بدین ترتیب برای هر مولفه ی "دید" یک مولفه ی "کنترل" و یک مولفه ی"مدل" کلی نیز، برای تمام سیستم در نظر گرفته شده است. در واقع مشتریان را به عنوان یکی از دیدهای سیستم در نظر می گیریم که به آن یک مولفه کنترل نسبت داده و مولفه ی مدل نیز آن را مدیریت می کند.

شکل ۲ ساختار سیستم نظرسنجی مشتریان را با کارت‌های CRC براساس الگوی طراحیMVC نشان می دهد.

همانگونه که در مطالب بخش های قبل گفته شد در شکل ۳ کارت‌های CRC به نمودار مولفه تبدیل شده است که کلیشه قابلیت اطمینان به مولفه های آن الحاق شده است. درآخر شکل ۴ یک مدل قابل اجرا از ساختار الگوی طراحی MVC با استفاده از شبکه پتری رنگی را نشان می دهد.که قادر به محاسبه قابلیت اطمینان هر یک از مولفه‌های سیستم می باشد. جدول ۱ نشان می دهد که در هر گام از اجرا قابلیت اطمینان هر مولفه به چند درصد می رسد این محاسبه تا چهار گام پیش رفته و سپس میانگین قابلیت اطمینان برای هر مولفه نیز محاسبه شده است.



Model

Responsibilities:

Name	Collaborator
notity	CustomerController
updateview	CustomerView
updatecontroller	CustomerController
getdata	CustomerView,CustomerController

CustomerView

Responsibilities:

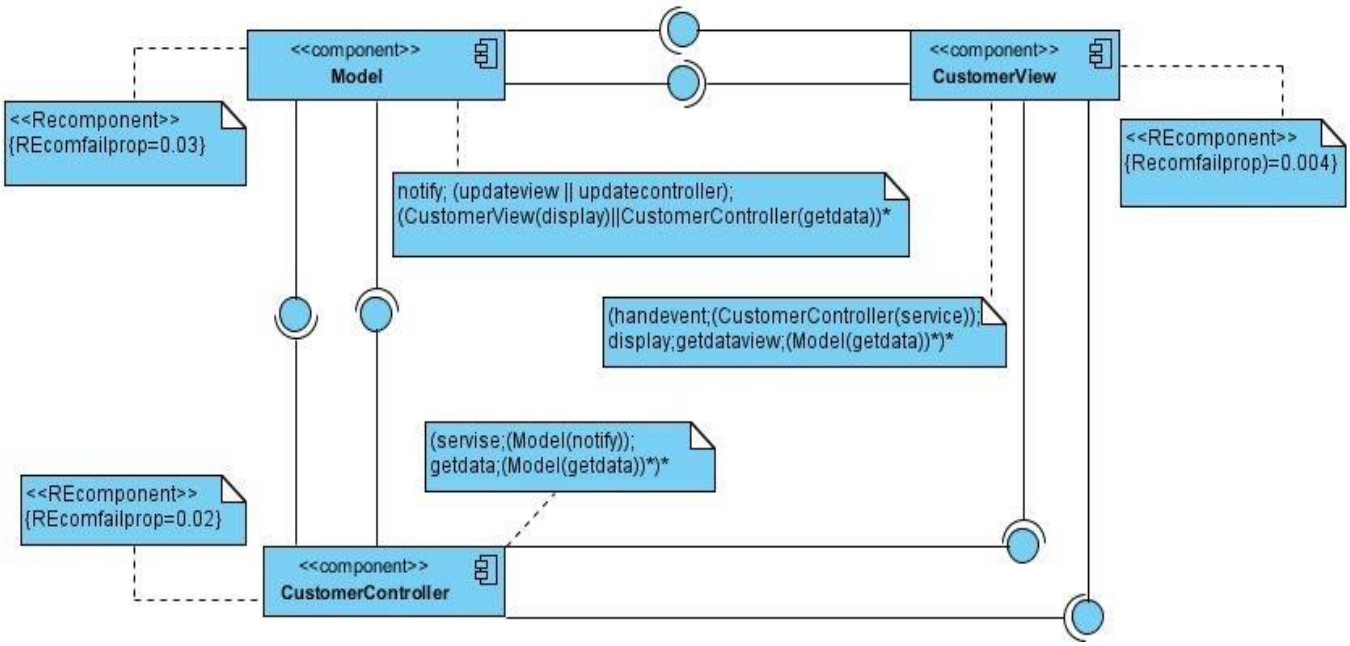
Name	Collaborator
handevent	CustomerController
display	Model
getdataview	Model

CustomerController

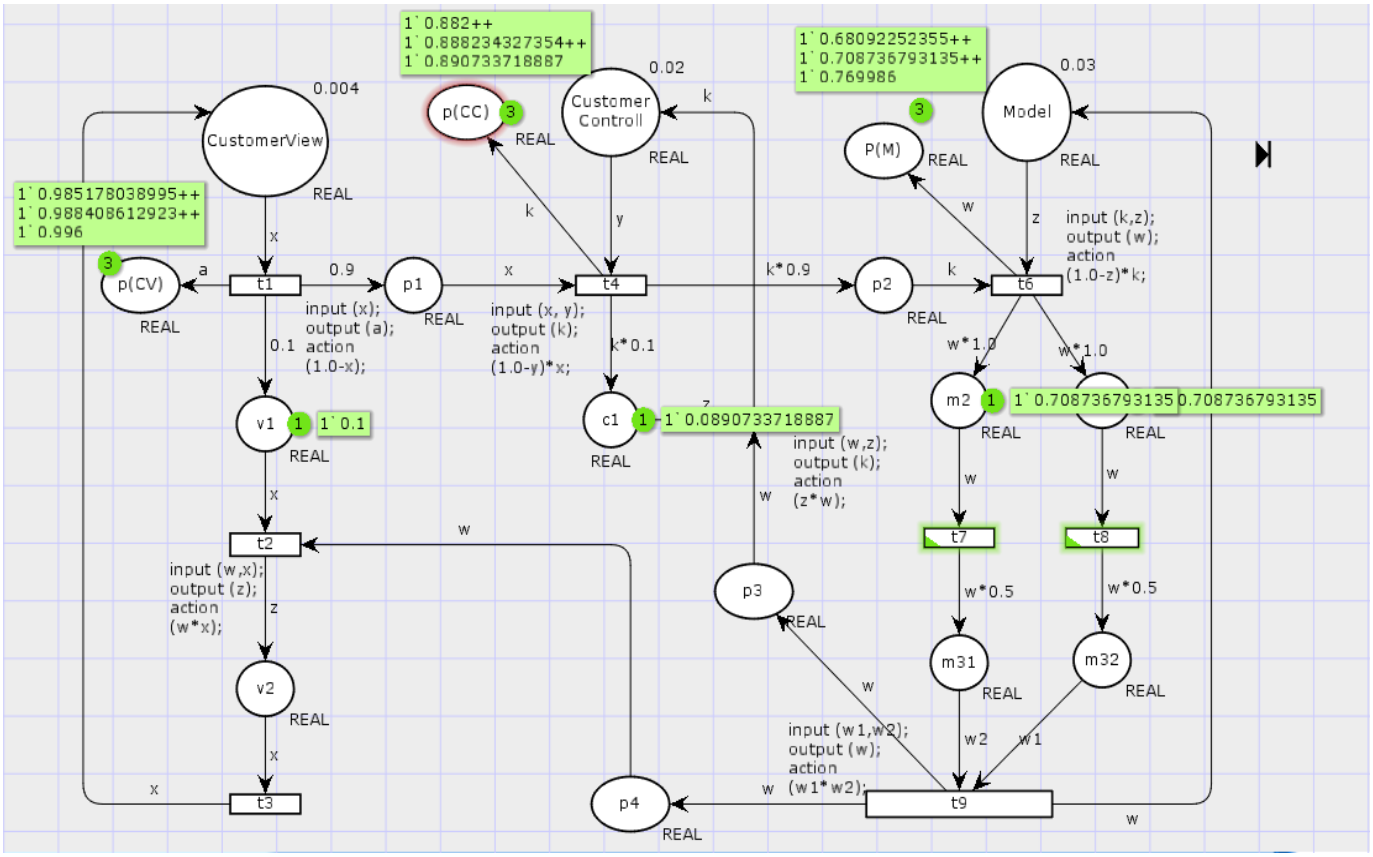
Responsibilities:

Name	Collaborator
service	Model
getdata	Model

شکل ۲: نمایش بخشی از ساختار سیستم نظرسنجی مشتریان با کارت CRC



شکل ۳: تبدیل کارت CRC به نمودار مولفه به همراه کلیشه قابلیت اطمینان



شکل ٤: شبکه پتری حاصل از نمودار مولفه جهت محاسبه قابلیت اطمینان هر مولفه

جدول۱: قابلیت اطمینان برای هر یک از مولفه

گام ها	قابلیت اطمینان مولفه Model	قابلیت اطمینان مولفه Customer Controller	قابلیت اطمینان مولفه Customer View
مرحله اول	٪۰,۷۶۹	٪۰,۸۹۰	٪۰,۹۹۶
مرحله دوم	٪۰,۷۰۸	٪۰,۸۸۹	٪۰,۹۸۸
مرحله سوم	٪۰,۷۰۰	٪۰,۸۸۸	٪۰,۹۸۷
مرحله چهارم	٪۰,۶۸۰	٪۰,۸۸۲	٪۰,۹۸۵
متوسط قابلیت اطمینان	٪۰,۷۱۴۲	٪۰,۸۸۷۲	٪۰,۹۸۹

۶. نتیجه گیری

معماری نرم‌افزار اولین فرایند توسعه نرم‌افزار است که از مراحل رسیدن به ارزیابی ویژگی های کیفی به حساب می آید. در این مقاله نیز با انتخاب الگوی معماری MVC و ارزیابی قابلیت اطمینان مولفه‌های آن، میتوان مولفه‌ای که باعث ایجاد شکست در سیستم خواهد شد را شناسایی نمود.

در ادامه، به عنوان کارهای آینده پیشنهاد شده که از نمودار های دیگر زبان مدلسازی یکنواخت مانند استقرا نیز جهت ارزیابی قابلیت اطمینان استفاده گردد. همچنین می توان با ترکیب با نمودارهای همچون مورد کاربری و ترتیب که رفتار سیستم را نشان می دهد قابلیت اطمینان را محاسبه نمود. همچنین به عنوان کار دیگر می توان از دیگر سبک ها و الگو های معماری نیز استفاده کرد و ساختار و رابطه بین مولفه های آنها را با این کارت‌ها نمایش داد و سپس نیز آنها را ارزیابی نمود.

منابع

١. Balsamo, S., Marco, A. D., Inverardi, P., & Simeoni, M. (٢٠٠٤). Model-Based Performance Prediction in Software Development: A Survey. *TRANSACTIONS ON SOFTWARE ENGINEERING, Vol.*٣٠, *No.*٥, PP. ٢٩٥-٣١٠ .

٢. Zhang, C. and Shen, W.(٢٠١٢). The Studies on Integrated Development Approach Based on the Architecture and MVC Design Pattern, in Electrical & Electronics Engineering Symposium on (EEESYM). on: Kuala Lumpur p. ٥٧٨ - ٥٨٠.

٣. Emadi, S. and Shams, F.(٢٠٠٩). a new executable model for software architecture based on petri net. Indian Journal of science and technology, ٢(٩): p. ١٥-٢٥.

٤. Bass, L., Clements, P., & Kazman, R. (٢٠١٢). *Software Architecture in Practice*, ٣th Edition, Addison Wesely, the United States.

٥. Buschmann, F., Schmidt, D., Stal, M & ,.Rohnert, H. (٢٠٠١). *Pattern-Oriented Software Architecture a System of Pattern*, ١th Edition, England: John'Wiley & Sons Ltd, England.

٦. Majidi, F., & HarounAbadi, A. (٢٠١٥). Presentation of an executable model for evaluation of software architecture using blackboard technique and formal models. *Journal of Advanced Computer Science & Technology,Vol.* ٤, *No.*١, PP. ٢٣-٣١

٧. Cristescu, P.M. , Stoica, E.A. & Ciovică, L.V. (٢٠١٥). The Comparison of Software Reliability Assessment Models, ٢٢nd International Economic Conference of Sibiu ٢٠١٥, IECS ٢٠١٥ "Economic Prospects in the Context of Growing Global and Regional Interdependencies, PP٦٦٩-٦٧٥.

٨.Abbasabadi, S. & Motemani, H.(٢٠١٢). Software reliability evaluation using stochastic Petri net, International Conference on nonlinear modeling and optimization, Amol, University of North, PP.

٩. Abdolmaleki, H., Razazi, M.R. & Farahi, A.(٢٠١١). Providing a way to evaluate the reliability of software-based architecture combines styles such as pipes and filter by mapping layer and Markov chain model Petri net, The first Conference of artificial intelligence systems and their applications, Tehran, Payam Noor University of Tehran,PP.١-٩.

۱۰.Soleimani Nisiani, B., Velayati, S., Bahadorpour, M., Safari, Z. & NematBakhsh, N.(۲۰۱۱). Introduced an approach for identifying software reliability model, The Secend Conference on Reliability Engineering, PP.۱-٦.

۱۱. Rodrigues, G., Rosenblum, D. & Uchitel, S. (۲۰۰۵),Using Scenarios to Predict the Reliability of Concurrent Component Based Software System, Proceedings of the ۸th international conference, held as part of the joint European Conference on Theory and Practice of Software conference on Fundamental Approaches to Software Engineering, PP.۱۱۱-۱۲٦.

۱۲. Kotaiah, B., Prasad , M.V.S. & Khan, R.A. (۲۰۱۵). An Analysis of Software Reliability Assessment with Neuro-Fuzzy based Expert SystemsProceedings of the International Conference on Soft Computing and Software Engineering. PP.۲٤۱-۲۵۲.

۱۳. Lou, J., Jiang ,Y., Shen, Q., Shen, Z., Wang, Z. & Wang, R. (۲۰۱٦). Software reliability prediction via relevance vector regression, Neurocomputing Vol.٦, No.۱, PP.۱٥٤-۱٦۲

۱٤. Chong,Ch.Y. & Lee S.P.(۲۰۱٥). Analyzing maintainability and reliability of object-oriented software using weighted complex network, Journal of Systems and Software, Vol.۱۱, No.۱, PP.۲۸-٥۳.

SCSE'۱٥), PP.۹۲–۹۸.

۱٥. Motameni, H., & Nemati, M. (۲۰۱٤). Mapping CRC Card into Stochastic Petri Net for Analyzing and Evaluating Quality Parameter of Security. *IJE TRANSACTIONS B: Applications, Vol. ۲۷, No.*٥, PP. ٦۸۹-٦۹۸..

۱٦. Shamsaliee, f., *modelin the behaviour of processes using collaborating objects*, in *department of computer science*. ۱۹۹٦, university of manchester. p.۲۲۰.