ارائه روشی جهت استفاده از معماری نرم افزار در آزمون سطح کد

طاهر نوحی[۱] ، سید مهران شرفی[۲]

۱)دانشجوی مقطع کارشناسی ارشد مهندسی کامپیوتر نرم افزار(t.noohi@gmail.com)

۲)استادیار دانشگاه آزاد واحد نجف آباد (me_sharafi@yahoo.com)

چکیده

این تحقیق در رابطه با استفاده از معماری نرم‌افزار به عنوان یک مدل مرجع برای تست میزان تبعیت یک سیستم پیاده‌سازی شده با خصوصیات معماری آن است. خصوصیات SA[۱] dynamics را برای شناسایی یک طرح مفید در تعاملات بین اجزای سیستم و انتخاب کلاسهای تست نظیر به نظیر با رفتارهای معماری وابسته استخراج می‌کنیم. SA dynamics با LTS[۲] ها مدل می‌شود . رهیافتی که مرکب از LTS های مناسب است به اختصار ALTS[۳] نامیده می‌شوند. ALTS ها نمادهای ویژه‌ای از SA dynamics با تمرکز بر خصوصیات مربوط مختصرسازی با حذف موارد غیرقابل توجه را پیشنهاد می‌دهد. مسلما نتیجه گرفتن از یک مجموعه کلاس تست مناسب مستلزم نتیجه گرفتن از یک مجموعه مسیرهایی است که به طور مناسب ALTS را پوشش می‌دهد. یک رابطه بین خلاصه‌ای از تست‌های SA و مهم‌تر از آن تستهای اجرایی باید برقرار شود که تستهای معماری به وجود آمده می‌توانند به تست‌های در سطح کد برسند.

کلمات کلیدی:

معماری نرم افزار،تست نرم افزار،مهندسی نرم افزار ، استراتژی تست

۱)مقدمه

در سالهای اخیر، مطالعه معماری نرم‌افزار به عنوان یک راه حل و قانون مستقل با راهکارها و روشها و ابزار مخصوص به خود ظاهر شده است[1] . SA رهیافت‌های امیدوار کننده‌ای که می‌توان طراحی و تحلیل سیستمهای توزیع شده پیچیده را به کمک آنها انجام داد و مشکلات مربوط به بزرگ شدن مسئله‌ها را به آن مهار کرد ، ارائه می‌دهد.

[1] - Software Architecture

[2] - Labaled Transition System

[3]- suitable LTS abstractions

چگونگی استخراج SA Dynamic ها ونحوه مدل کردن وبه دست آوردن کلاسهای تست مناسب در حال حاضر جز جنبه های مجهول این مساله می باشد

به عبارت دیگر SA ابزاری فراهم می‌کند تا به واسطه آن کاربردها و مسائل بزرگتر قابل مدیریت باشد. SA با پشتیبانی از روشهای مدل سازی رسمی یک سیستم، امکان توصیف توپولوژیکال (استاتیک)و توصیف رفتاری (دینامیک) سیستم را فراهم میکند. SA سیستم را با شرایط مربوط به اجزای[1] آن و ارتباط دهندگان[2] آن مدل می‌کند. که اجزا خلاصه ای از زیر سیستم ها را نشان می‌دهند و ارتباط دهندگان تعامل بین اجزا را انجام می‌دهند. هم در بعد صنعت و هم در بعد آموزشی از SA در جهت بهبود بخشیدن به قابلیت اعتماد بر سیستم‌های پیچیده به طور فعال استفاده می‌شود.

سخت‌ترین قسمت از فرآیندتوسعه، پروسه تست است. تست نرم‌افزار شامل اعتبارسنجی دینامیک از رفتارهای برنامه که از مشاهده اجرای برنامه روی مجموعه‌ای از نمونه تستها به دست می‌آید، می‌باشد

چندین محقق مزایای استفاده از روشهای رسمی در تست را به خوبی توضیح داده‌اند و چندین تکنیک برای انتخاب تست‌ها از روشهای نیمه رسمی و جبری [2] و همچنین روشهای بر پایه مدل ارائه شده است . اخیراً رهیافتهای مختلفی در زمینه تست پیشنهاد شده که اساس آنها خصوصیات معماری می‌باشد.[3] در همین راستا می‌توانیم مشکل شناسایی کلاسهای تابعی تست مناسب برای انجام تست در سیتسم‌های پیچیده جهان واقعی را بیابیم. [5][4]

هدف ما فراهم کردن یک مدیریت کننده تست با یک روش سیستماتیک برای نشان دادن کلاسهای تست مناسب برای تست‌های رده بالا و پالایش آنها برای بدست آوردن یک تست محکم در سطح کد است. این رهیافت برپایه خصوصیات SA dynamics است که برای شناسایی یک نمای مناسب ازتعاملات بین اجزای سیستم و انتخاب کلاسهای تست مشابه با رفتارهای معماری متناظر آن استفاده می‌شود. یک تست معماری باید همانطور که اجرا می‌شود به یک تست در سطح کد برسد. در پایان گامی را پی می‌گیریم تا با یک روش دستی اقدام به یافتن یک رابطه رسمی بین توضیحات SA و کد بیابیم.

نکاتی راجع به تفاوتهای روش‌های استفاده شده ، برای مثال به کار بردن یک معیار جهت به دست آوردن یک مسیر خاص با دربرگیری مطلوب مساله در برابر استفاده از LTS کامل .

[1] - Component

[2]- Connector

حقیقتاً ALTS ابزاری است برای فهمیدن اینکه چه چیزی باید تست شود و به عبارتی دیگر ALTS یک مدلی از سیستم را آماده می‌کند در حالی که یک معیار پوشش[1] یک روش کارآمد تولید یک مجموعه از سناریوهای تست از یک مدل ویژه است. این مقاله شامل دو قسمت اساسی می‌شود : ۱) توسعه یک روش برای استخراج مشخصات آزمون مربوط در یک سطح معماری

۲) تست قابلیت تداوم رهیافت در یک محیط واقعی که مسلما بررسی تجربی روی یک نمونه مطالعاتی برای این هدف مفیدتر خواهد بود.

۲)مروری بر روش ارائه شده

چندین زبان توصیف معماری برای مدل کردن SA dynamics به LTS ها استفاده می‌شوند. در اکثر ADL[2] ها از LTS برای مدل سازی SA استفاده می شود. که می‌تواند از یک تشریح باتمام جزئیات مثلاً حالات وانتقال‌های مربوط که در متن SA dynamics وجود دارد، مشتق شود.

مدل LTS یک پنج‌تایی (S,L,S_0,S_f,T) است که S مجموعه حالات، L شامل مجموعه‌ای از برچسب‌های مهم یا همان عملیات که بر مبنای الفبای مربوط به LTS می‌باشد.

S_0 یک ازحالات موجود در S $(S_0 \in S)$ می‌باشد که تحت عنوان حالت اولیه نامیده می‌شود S_F زیرمجموعه‌ای از حالات $(S_F \subseteq S)$ می‌باشد که حالات نهایی نامیده می‌شود. و T یک انتقال با یک عنصر یا عمل مربوط است که در L قرار دارد به عبارت دیگر یک انتقال با برچسب L ، $T = \{\xrightarrow{L} \subseteq S \times S \mid L \in L)$.

در ادامه بحث عناوین Lables و Action به جای هم استفاده خواهند شد. مسیر P که به صورت $P = S_0 \xrightarrow{L} S_1 \xrightarrow{L_2} S_2 \xrightarrow{L3} \ldots \xrightarrow{Ln} Sn$ تعریف می‌شود یک مسیر کامل است اگر S_0 حالت اولیه و S_n حالت نهایی می‌باشد. همچنین برای رعایت اختصار می‌توان از دنباله‌ای از برچسب‌ها نیز استفاده کرد.

$$P = L_1 L_2 \ldots, L_n$$

راه حل ارائه شده برای مساله مطرح شده شامل ۴ گام منطقی می باشد که عبارت است از:

[1] -Coverage criterrion

[2] - Architectural Description Languages

تعریف obs-function های گوناگون بوسیله بررسی SA Dynamic هااز دید گاههای مختلف که هر کدام از obs-function باید یکی از موارد مهم و حیاتی را در تست تشریح نماید.

۲) بکار گیری obs-function تولید شده در گام اول در LTS و تولید یک ALTS که خلاصه ای از LTS بوده و نشان دهنده رفتارهای مهم و جالب LTS با توجه به دیدگاه انتخاب شده می باشد.

۳) در گراف بدست آمده در گام ۲ (ALTS)که به نسبت LTS اولیه و اصلی قابل مدیریت تر است معمار نرم افزار مسیری روی ALTS که شامل قالب مهمی از رفتار می باشد برای تست انتخاب می کند

۴)در نهایت تستهای سطح بالای تولید شده به تست کننده نرم افزار ارسال میگردد و او با اجرای تست مشاهدات خود را در مورد تبیعت پیاده سازی جاری از مدل معماری ثبت میکند.

در صورتیکه فعل و انفعالات تشریح شده در سطح معماری در سطح پیاده سازی نمود و اجازه اجرا نداشته باشد تطابق بین مدل معماری و پیاده سازی تایید نمی گردد

گامهای ۱ تا ۳ روشی برای استخراج تست های معماری از خصوصیات معماری نرم افزار میباشد و در گام چهارم اقدام به اجرای تست ها در سطح کد خواهیم نمود

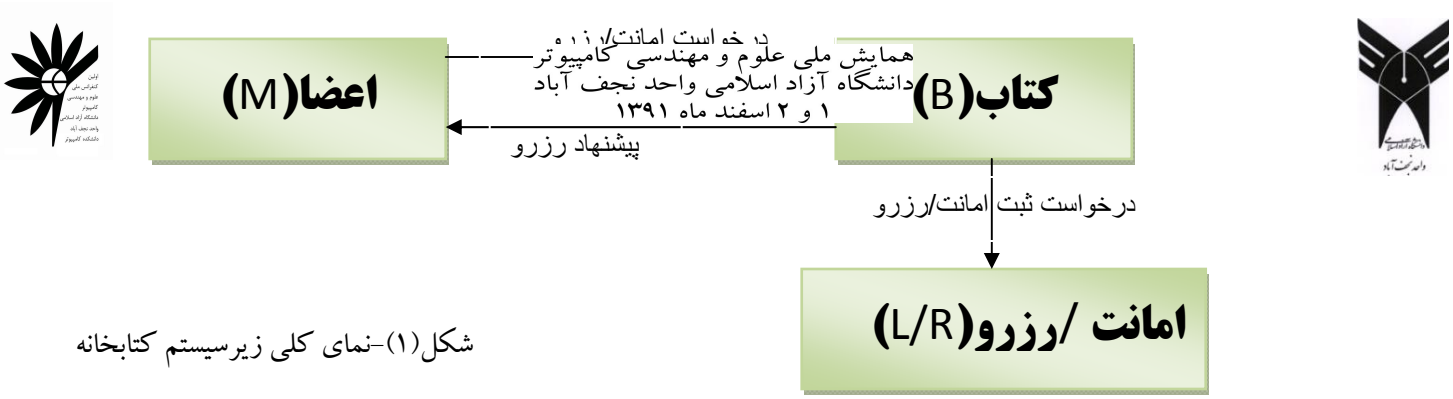
در این مقاله نمی توانیم به یک پروسه استخراج رسمی دقیق از SA به کد استفاده نماییم.که البته این موضوع در مورد گام چهارم محسوس تر است و مسلما در این گام از کمترین درجه رسمی بودن استفاده می شود.در نتیجه ی بکار گیری این رهیافت در یک مورد مطالعاتی ،نمیتواند دلیلی استقرایی بر جوابگو بودن این روش در تمامی سیستمها باشدو فقط می تواند به درک مفاهیم اولیه در مورد اجرای تست های سطح معماری در سطح کد ،کمک کند.

۳)به کار گیری روش

برای مشخص شدن نحوه عملکرد رهیافت ، بهتر است این رهیافت به صورت گام به گام روی یک سیستم اعمال گردد تا ضمن افزایش درک از روش ، کارایی آن نیز مشخص گردد.

سیستم امانت ورزرو کتاب برای اعضای هر سازمان ، می تواند به عنوان یک زیر سیستم مهم در یک سیستم جامع تلقی شود ،لذا در این مقاله با توضیح روش بر روی این زیرسیستم به تشریح راه کار می پردازیم.

ابتدا نمای کلی از این زیر سیستم باید نمایش داده شود(شکل ۱)



شکل(۱)-نمای کلی زیرسیستم کتابخانه

با توجه به نمای فوق(شکل ۱) می توان LTSها را استخراج نمود که به شرح زیر می باشد:

M.Send Loan Request-To-**B**

B.Check Book Status

B.Send Loan Registeration-To-**L/R**

B.Send Reserve Promotion-To-**M**

M.Send Reserve Request-To-**B**

B.Send Reserve Registeration-To-**L/R**

در مرحله بعد باید LTS ها خلاصه شده و LTSهای مهمتر از آن استخراج شود و در واقع LTSها به دو دسته مرتبط(R) و غیر مرتبط (R′) تقسیم شوند که دامنه LTSهای مرتبط D و غیر مرتبط T می باشد.گروه مرتبط که همان LTSهای عضو D می باشند منجر به تشکیل ALTS می گردد.

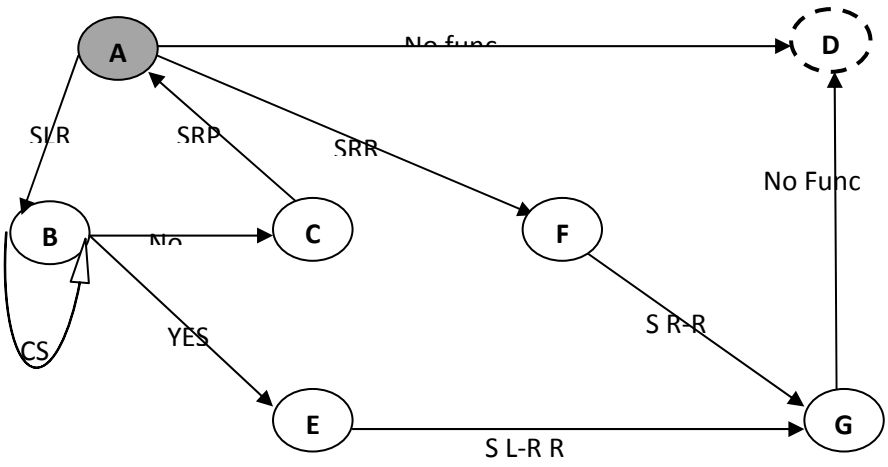
D={Send Loan Req,Send Res Req,Send Res Prom,Check Stat,Send Loan-Req Req ,Send Res-Reg Req}
obs(M.Send Loan Request-To-**B)=** Send Loan Req(SLR)
obs(B.Check Book Status**)=** Check Stat(CS)
obs(B.Send Loan Registeration-To-**L/R)=** Send Loan-Req Req(S L-R R)
obs(B.Send Reserve Promotion-To-**M)=** Send Res Prom(SRP)
obs(M. Send Reserve Request-To-**B)=** Send Res Req(SRR)
obs(B. Send Reserve Registeration-To-**L/R)=** Send Res-Reg Req(S R-R R)
برای بقیه: obs(r)ϵ T

با توجه به حالات مختلفی که می تواند وجود داشته باشد FSP[1] به شکل زیر(شکل ۲) خواهد بود

[1] -Finite State Process

S_0 :A

S_F=D



شکل (۲)-ALTS مربوط به سیستم کتابخانه

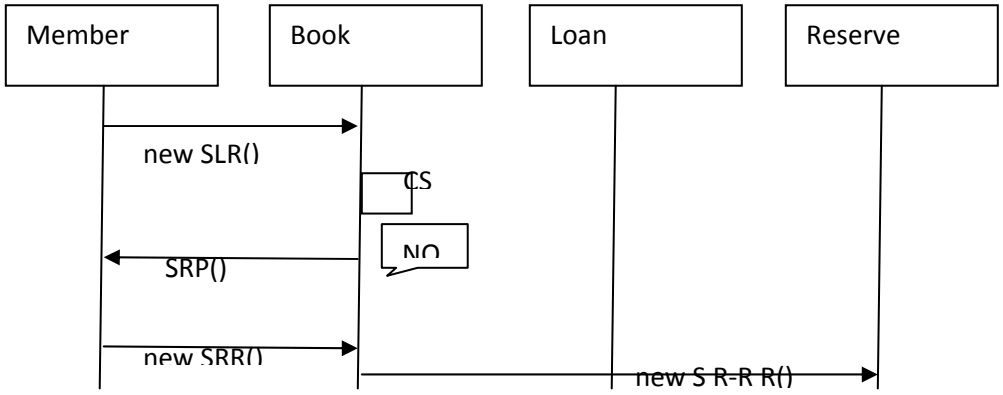
حال می توان با استفاده از ALTS موجود و مسیرهای آن سناریوهای تست را براحتی بدست آورد ، مسیر ها عبارتند از:

مسیر ۱:**ABCAD** کاربر درخواست امانت کتاب میدهد ،کتاب برای امانت موجود نیست و پیشنهاد رزرو به کاربر داده میشود ولی کاربر بدون رزرو از سیستم خارج می شود

مسیر ۲: **ABCAFGD** کار بر درخواست امانت کتاب میدهد ،کتاب برای امانت موجود نیست و پیشنهاد رزرو به کاربر داده میشود و کاربر درخواست رزرو میدهد و کتاب برای او رزرو میشود

مسیر ۳: **ABEGD** کاربر درخواست امانت میدهد، کتاب موجود است و امانت او در سیستم ثبت می شود

حال با توجه به سناریو های تست بدست آمده میتوان این سناریو تست را در پیاده سازی انجام شده نیز اجرا کرد، بنابراین با توجه به اینکه در سناریوهای بدست آمده اجزای M,B ,L/R مشارکت دارند میتوان نمودار توالی زیر(شکل ۳) را به عنوان یکی از سناریو های تست سطح کد در نظر گرفت.



شکل ۳-پیاده سازی رزرو کتاب

۴)نتیجه گیری

با توجه به راه کار ارائه شده می توان سیستم های پیچیده را با معماری دقیق و استخراج کلاسهای تست سطح بالا و در نتیجه پالایش آنها ، برای تست سطح کد سناریو هایی را از قبل مشخص کرد تا گروه تست کننده با آگاهی بیشتر از فرآیندهای بحرانی اقدام به تست نموده و در نتیجه فرآیند تست دقیق تر و کم هزینه تر انجام شود.

۵)راه کار های آینده

در این مقاله نشان داده شد که توصیفات معماری تا چه حد میتواند به فرآیند توسعه و در نهایت تست نرم افزار کمک کند لذا در آینده میتوان با رسمی سازی روش ،فر آیند استخراج LTS و خلاصه سازی آن در سیستمهای توزیع شده پیچیده را تاحدی تسهیل و به دور از اشتباهات انسانی نمود.

٦)منابع

[1] L. Bass, P. Clements, and R. Kazman, “ Software Architecture in Practice”. Addison-Wesley, 1998.

[2] T.J. Ostrand , M.J. Balcer, “The Category-Partition Method for Specifying and Generating Functional Tests” Comm. ACM, vol. 31, no. 6, pp. 676-686, June 1988.

[3] M.J. Harrold, “Testing: A Roadmap,” Proc. ACM ICSE 2000 Conf., The Future of Software Eng., A. Finkelstein, ed., pp. 61-72, 2000.

[4] A. Bertolino, F. Corradini, P. Inverardi, and H. Muccini, “Deriving Test Plans from Architectural Descriptions,” ACM Proc. Int’l Conf. Software Eng., pp. 220-229, June 2000.

[5] A. Bertolino, P. Inverardi, and H. Muccini, “An ExplorativeJourney from Architectural Tests Definition downto Code Tets Execution,” IEEE Proc. Int’l Conf. Software Eng., pp. 211-220, May 2001.

[6] Henry Muccini, Antonia Bertolino, and Paola Inverardi. “Using Software Architecture for Code Testing,” IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 30, NO. 3, MARCH 2004