اصفهان دانشگاه
دانشکده فنی و مهندسی
گروه کامپیوتر

پایان نامه کارشناسی ارشد رشته مهندسی کامپیوترگرایش نرم افزار

ارایه روشی برای تولید خودکار موارد تست با استفاده از ویوچارت ها

استاد راهنما:
دکتر ناصر نعمت بخش

استاد مشاور:
دکتر بهروز ترک لادانی

پژوهشگر:
مجتبی وطنی

مهرماه ۱۳۸۶

دانشگاه اصفهان
دانشکده فنی و مهندسی
گروه کامپیوتر

پایان نامه کارشناسی ارشد رشته مهندسی کامپیوتر
گرایش نرم افزار آقای مجتبی وطنی

تحت عنوان

ارایه روشی برای تولید خودکار موارد تست نرم‌افزار با استفاده از ویوچارت ها

در تاریخ ۲۵ / ۷ /۸۶ توسط هیأت داوران زیر بررسی و با درجه خوب به تصویب نهایی رسید.

۱-استاد راهنمای پایان نامه دکتر ناصر نعمت بخش با مرتبه ی علمی استادیار امضا

۲-استاد مشاور پایان نامه دکتر بهروز ترک لادانی با مرتبه ی علمی دانشیار امضا

۳-استاد داور داخل گروه دکتر ناصر قاسم آقایی با مرتبه ی علمی استاد امضا

۴- استاد داور خارج از گروه دکتر حسن ابوالحسنی با مرتبه ی علمی دانشیار امضا

امضای مدیر گروه

سپاس و ستایش مر خدای را جل و جلاله

که آثار قدرت او بر چهره روز روشن، تابان است

و انوار حکمت او در دل شب تار، درفشان

آفریدگاری که خویشتن را به ما شناساند

و درهای علم را بر ما گشود

و عمری و فرصتی عطا فرمود تا بدان

بنده ضعیف خویش را در طریق علم و معرفت بیازماید

تقدیم به:

روح پاک پدرم

که عالمانه به من آموخت

تا چگونه در عرصه زندگی

ایستادگی را تجربه نمایم

و به مادرم

دریای بی کران فداکاری و عشق

که وجودم برایش همه رنج بود

و وجودش برایم همه مهر

چکیده

یکی از مهم‌ترین مسائلی که طراحان نرم‌افزار با آن روبرو هستند، تست نرم‌افزار است. تست نرم‌افزار با استفاده از ابزارهای پیش از کد نظیر مدل و مشخصات به توسعه‌دهندگان نرم‌افزار این امکان را می دهد که دنباله‌های تست را پیش از تولید کد و به موازات آن تولید کنند. به این روش، تست مبتنی بر مدل می‌گویند.

روش‌های زیادی برای این منظور ارائه شده‌اند که از مدل‌هایی نظیر ماشین های حالت متناهی یا مدل‌های فرمال برای تولید خودکار موارد تست استفاده می کنند. بدنبال کاستی‌هایی نظیر تولید موارد تست طولانی و تولید موارد تست زائد، هدف این پروژه ارائه روش جدیدی برای این منظور می‌باشد.

در این روش از مدل ویوچارت ها استفاده شده است. با این روش پس از مرحله تحلیل نیازمندی‌ها، سیستم به صورت دیاگرام‌های ویوچارت مدل می‌شود و مدل ایجاد شده به دو گروه تولیدکنندگان کد و موارد تست تحویل داده می‌شود. این دو گروه به صورت موازی مشغول به کار می‌شوند. با پیشرفت تولید کد، موارد تست متناسب تولید می‌شوند و بخش های مختلف تست می‌شوند. این ویژگی از جزء به جزء بودن نمودار ویوچارت ها سرچشمه می‌گیرد. به این ترتیب سیستم به صورت پی در پی توسط موارد تست کوچک تست می‌شود. بااستفاده از این روش تولید تست، به دنباله‌های تستی دست می‌یابیم که کوتاه و جامع می باشند. به این ترتیب اجرای دنباله‌های تست زمان زیادی نمی‌برد و از طرفی به دلیل کوتاهی آنها و وابستگی آنها به بخش خاصی از برنامه، منبع خطا به راحتی یافت می‌شود.

کلید واژه ها : تست نرم افزار ، تست مبتنی بر مدل، مورد تست ، تولید خودکار، ماشین حالت متناهی ، ویوچارت

فهرست مطالب

عنوان **صفحه**

عنوان **صفحه**

عنوان **صفحه**

فهرست شکل‌ها

عنوان **صفحه**

عنوان **صفحه**

فهرست جدول‌ها

عنوان **صفحه**

فصل اول

توصیف کلی تحقیق

۱-۱- موضوع تحقیق

هر روزه بر تعداد شرکت ها و مؤسساتی که سیستم های خود را به صورت مکانیزه و خودکار در می‌آورند افزوده می‌شود. به این ترتیب با افزایش تمایل مؤسسات به خرید و نصب نرم افزارهای کامپیوتری، نرم‌افزارها به تدریج در بخش های حیاتی و مهم تری از جوامع به کار گرفته می‌شوند و این موضوع نیاز به توجه و تمرکز در امر اطمینان نرم افزارها را افزایش می‌دهد؛ چنانکه یک خطای کوچک در بخش های حیاتی مانند بانک ها یا نیروگاه ها می‌تواند صدمات و خسارت های شدید و جبران ناپذیری را به بار آورد.

یکی از مهم ترین مسایل در مقوله قابلیت اطمینان نرم افزارها، تست آن است. در هر شرایطی که نرم افزار نوشته شده باشد یا هر چقدر گروه سازنده و برنامه نویس قوی بوده باشند، خطاهای نرم افزاری، اجتناب ناپذیرند.

در مرحله تست نرم افزار، نرم افزار تست می‌شود تا خطا هایی که در طراحی و ساخت آن به طور ناخواسته بوجود آمده اند را نمایان کند. یک استراتژی معمول برای تست نرم افزار، روش های طراحی موارد تست نرم افزار را به صورت دنباله ای از گام های خوش تعریف در می‌آورد که به تولید موفق نرم افزار می‌انجامد. مسئله مهم در انتخاب یک مورد تست، چگونگی تعیین یک دنباله از موارد تست است، به گونه ای که به طور مؤثر عملکرد مورد انتظار سیستم را نشان دهد. در ضمن، یک مورد تست باید کوتاه باشد و تا حد امکان بتواند تمامی عیب هایی که ممکن است در سیستم رخ دهند را نشان دهد.

روش های موجود انتخاب تست در برخورد با این دو مورد و میزان فرمال بودن، به گونه هایی متفاوت عمل می‌کنند و هدف اصلی را یکی از این موارد انتخاب می‌کنند.

در کل استراتژی‌های تولید موارد تست را می‌توانیم به دو گروه تقسیم کنیم:[1]

۱- براساس کد برنامه

۲- بر اساس ابزارهای پیش از کد، نظیر نیازمندی‌ها، مشخصات و مدل‌های طراحی

انتخاب موارد تست از روی کد برنامه، علاوه بر پیچیدگی ذاتی خود، مستلزم انتظار تا پایان کار است و در کل، دوره ساخت و تحویل نرم‌افزار را افزایش می‌دهد. اما استفاده از ابزارهای پیش از کد این انتظار را حذف می‌کند و می‌توان همگام با تولید کد، موارد تست را از نیازمندی‌ها استخراج کرد. اما در بین ابزارهای پیش از کد دو مورد اول یعنی نیازمندی‌ها و مشخصات، جزیی و غیر فرمال هستند و بنابراین استخراج موارد تست از آنها مشکل است و موارد تست استخراجی نیز طبعاً ناکامل هستند. از طرفی در طول فرآیند مهندسی نرم‌افزار، نیازمندی‌ها و مشخصات، مدل طراحی را شکل می‌دهند که در تولید کد نرم‌افزار استفاده می‌شود. به این ترتیب مدل طراحی کامل ترین ابزار پیش از کد است.

۱-۱-۱-تکنیک‌های توصیف مدل رفتاری سیستم/کاربر

در کل تکنیک های توصیف مدل رفتاری سیستم/کاربر را می‌توانیم مطابق زیر تقسیم بندی کنیم[2]:

۱- جدول‌های تصمیم : که مجموعه‌ای از زوج‌های شرایط-عمل هستند.

۲- گرامرها

۳- زنجیره‌های مارکوف: یک فرآیند گسسته و تصادفی که بودن فرآیند در یک زمان خاص در یک حالت خاص مستقیماً به حالت قبلی آن وابسته است.

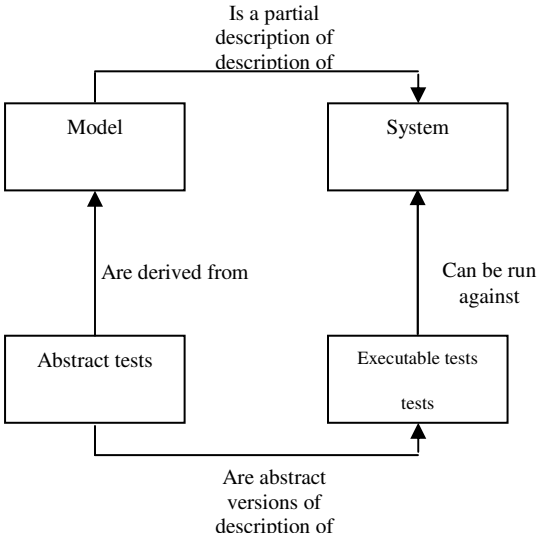
۴- استیت چارت ها[1]

جدول‌های تصمیم به خاطر موردی و غیرفرمال بودن ابزار مناسبی برای کار ما نیستند. گرامرها نحو برنامه‌نویسی را توصیف می‌کنند و بنابراین استخراج موارد تست از آنها پیچیدگی استخراج از کد برنامه را دارد و زنجیره‌های مارکوف هم با ساختار ریاضی، پیچیدگی خاص خود را دارند. استیت چارت ها نمونه کامل‌تری از ماشین‌های حالت متناهی هستند. بیشتر روش‌های تولید خودکار موارد تست از استیت چارت ها بهره می‌برند؛ اما

[1] Statecharts

مشکلاتی دارند که در قسمت ۴-۱ توضیح داده شده است و این مشکلات با استفاده از ویوچارت ها مرتفع می‌شوند.

در این پروژه تمرکز بر روی تست مبتنی بر مدل[1] است. تست مبتنی بر مدل به معنای استخراج موارد تست از مدلی است که رفتار سیستم را نمایش می‌دهد. مدل‌ها برای فهم، تعریف و توسعه سیستم‌ها به کار می‌روند. استفاده از مدل‌های طراحی در انتخاب موارد تست، نه تنها به عمل تست ساختار می‌دهد، بلکه دیگر نیازی نیست تا منتظر تکمیل کد سیستم باشیم. برای این منظور ابتدا باید روشی را برای نمایش مدل انتخاب کنیم و سپس روشی را برای استخراج موارد تست از این مدل ایجاد کنیم. شکل ۱-۱ این چرخه را نشان می‌دهد:



شکل ۱-۱ چرخه تست سیستم مبتنی بر مدل[3]

با یک مدل فرمال و قابل تفسیر توسط ماشین، موارد تست را می‌توان به صورت خودکار استخراج کرد. مدل‌ها را می‌توان از یک توصیف فرمال یا از ابزارهای دیاگرامی تولید کرد.

ویوچارت ها بر اساس استیت چارت ها و با افزودن مفهوم دیدگاه[2] به آن شکل گرفته است[4]. مفهوم دیدگاه رفتاری[3] از یک سیستم نرم‌افزاری به معنای رفتار سیستم از یک دید خاص می‌باشد. مثلاً از دید یک مشتری[4] در یک شبکه برای یک نرم‌افزار شبکه. در مرحله تعیین نیازمندی‌های[5] سیستم، مهندس نیازمندی‌های نرم‌افزار با کاربران بالقوه سیستم مصاحبه می‌کند و انتظار هر کدام از سیستم را ثبت می‌کند و یا به عبارت دیگر

[1] Model-Based Testing
[2] View
[3] Behavioral view
[4] Client
[5] Requirements

هر کاربر دیدگاه خود از سیستم را بیان می‌کند. بنابراین نتیجه این مصاحبه، دیدگاه رفتاری کاربر است. سپس این نتایج در قالب ویوچارت ها به نمایش درمی‌آیند.

در این روش از تکنیک تقسیم و حل[1] برای استخراج موارد تست استفاده می‌شود. به این ترتیب که ابتدا سیستم را براساس انواع کاربران، که در حقیقت دیدگاه‌های مختلف سیستم هستند، در نظر می‌گیریم، پس از بررسی نیازمندی‌های سیستم از دید هر کدام از این کاربران، نمودارهای ویوچارت را برای هر کدام بدست آورده و سپس موارد تست را به موازات تولید کد برنامه، برای هر کدام از این ویوچارت ها استخراج می‌کنیم و در نهایت از کنار هم قرار دادن این موارد تست به مجموعه‌ای از موارد تست دست می‌یابیم که در صورت انجام درست کارها در مرحله تعیین نیازمندی‌ها، می‌توانیم مطمئن باشیم که تمامی مسیرها و دیدگاه‌ها تست شده‌اند.

۱-۲- زمینه و تاریخچه تحقیق

همواره ساده ترین و ابتدایی ترین روش برای تست نرم افزار، تست تصادفی بوده که بعد از کامل شدن نرم‌افزار به طور تصادفی با سیستم کار می‌شود و ورودی هایی تصادفی به آن داده می‌شود تا به تدریج خطاهای آن بروز کنند؛ اما در طول چند دهه اخیر کتاب ها و مقالات زیادی در ارتباط با کاربرد مدل ها در توسعه تست و تحلیل اطمینان نرم افزار تولید شده اند.

روش‌هایی که تا امروز در تست مبتنی بر مدل ارائه شده‌اند را می‌توانیم به سه دسته تقسیم کنیم[5]:

۱-تست مبتنی بر ماشین‌های حالت: دیاگرام‌هایی که تغییرات حالات ماشین را نشان می‌دهند مبنای تست قرار می‌گیرند. از ماشین‌های حالت متناهی استفاده می‌کنند و نیاز به تعیین کامل مشخصات سیستم دارند مانند: SRC[2]، Formal Semantic، Class Testing و روشهای فرمال که از FSM، LTS[3]، ASM[4] و IOLTS[5] و ... بهره می‌برند.

۲-تست متنی بر سناریو[6]: دیاگرام‌هایی که تقابل سیستم با کاربران یا سایر سیستم‌ها را نشان می‌دهند مبنای کار قرار می‌گیرند. در اینجا به دنبال یافتن متدها و روش‌هایی هستیم که فعالیت‌های تست را با توصیفات UML[7] هدایت کنند، مانند MSC[8] و Totem[9].

---

[1] Divide and Conquer
[2] Software Reduction Cost
[3] Labeled Transition System
[4] Abstract State Machine
[5] Input/Output LTS
[6] Scenario based testing
[7] Unified Modeling Language
[8] Message Sequence Charts
[9] Testing Object orinTEd systems with the unified Modeling language

۳-روش‌های ترکیبی که دو روش یک و دو را با هم ترکیب می‌کنند: سناریوها را به شکل ساختار یافته و به صورت استیت چارت ها استفاده می‌کنند، مانند UMLAUT[1] که توصیفات UML را به یک زبان میانی تبدیل می‌کند یا AGEDIS[2]، AML [3] و SCENT[4].

۱-۳- اهداف تحقیق

همان طور که گفته شد روش‌های موجود تست مبتنی بر مدل بیشتر از UML، SRC،ASM یا ابزارهای مشابه با آنها استفاده می‌کنند که همگی بر پایه استیت چارت ها می‌باشند، اما استیت چارت ها دو مشکل عمده دارند[4]:

۱-تعداد زیاد حالات: با افزایش و رشد خطی سیستم، تعداد حالات به صورت نمایی رشد می‌کند؛ این رشد باعث افزایشی شدید در تعداد حالات در سیستم‌های بزرگ[5] می‌شود و در نتیجه موارد تست پیچیده تر می‌شوند. با بهره‌گیری از ویو چارت ها می‌توانیم این پیچیدگی را کمتر کنیم.

۲-فضای نام عمومی: در استیت چارت ها هیچگونه کنترلی در ارتباط با میدان دید[6] (مکانیسم کنترل حوزه با اصطلاحاتی مانند اعلان[7]، حوزه[8] و تقید[9]) سیستم وجود ندارد. ولی در ویوچارت ها این مشکل حل شده است. مثلاً یک تغییر در سیستم ممکن است حالت سیستم را از یک دیدگاه تغییر دهد ولی از یک دیدگاه دیگر، حالت سیستم همان حالت قبلی باقی بماند. به این ترتیب می‌توان با استخراج موارد تست مرتبط با دیدگاه ها، موارد تست را کوچک تر و گویاتر انتخاب کرد.

مشکلات موجود در استیت چارت ها، تأثیر مستقیم در فرآیند تست مبتنی بر مدل دارند. چنانکه پیچیدگی و افزونگی که با رشد سیستم ایجاد می‌شود، عمل استخراج موارد تست را با مشکل مواجه می‌کند و از طرفی به علت عدم وجود یک مکانیسم کنترل حوزه، موارد تست ایجاد شده حالت عمومی دارند. در حالی که با استفاده از ویوچارت ها می‌توان با محدود کردن یک مورد تست به یک دیدگاه، از بزرگ شدن موارد تست جلوگیری کرد. به این ترتیب به جای داشتن موارد تست بزرگ و پیچیده، مجموعه‌ای از موارد تست کوچک داریم که علاوه بر اجرای راحت تر آنها، ردیابی مشکلات کشف شده نیز سریع‌تر می‌شود. از دیگر مزایای این روش

---

[1] UML All pUrposes Transformer
[2] Automated Generation and Execution of test suites for DIstributed component-based Software
[3] AGEDIS Modeling Language
[4] SCENario-based validation and Test of Software
[5] Large-Scale
[6] Visibility
[7] Declaration
[8] Scope
[9] Binding

جزیی[1] بودن مشخصات به کار رفته در ویوچارت ها می‌باشد. به این ترتیب می‌توان موارد تستی برای نمونه‌های اولیه[2] سیستم فراهم کرد.

۱-۴- اهمیت و ارزش تحقیق

در پروژه‌های تجاری نمی‌توانیم تصور کنیم که همواره مدل کاملی از سیستم وجود دارد. تنها فرض قابل قبول، داشتن نیازمندی‌های نیمه‌فرمال است(مانند آنچه در UML است). از طرفی باید موارد تست را هر چه زودتر و حتی زمانی که سیستم به صورت جزیی مدل شده است، تهیه کنیم. برای برآورده کردن این نیازها روش‌هایی پیشنهاد شده‌اند مانند UIT[3] که از اجماع بین روش‌های موجود به جواب می‌رسد که البته بسیار وقت‌گیر است و هزینه زیادی دارد. اما روشی که با استفاده از ویوچارت ها ارایه می‌شود، روش واحدی است که هزینه و زمان کمتری می‌برد.

به این ترتیب با استفاده از ویوچارت ها در تولید تست، موارد تست به صورت فرمال و مستقل از سایر بخش های فرآیند تولید نرم افزار و بلافاصله پس از جمع آوری اطلاعات مصاحبه با کاربران سیستم(که دیدگاه‌های سیستم را شکل می‌دهند)، تهیه می‌شوند که این امر سرعت خطایابی و تحویل نرم افزار را افزایش می‌دهد و مانع از هزینه‌های احتمالی بعدی می‌شود.

۱-۵- کاربرد نتایج تحقیق

با این روش پس از مرحله تحلیل نیازمندی‌ها، سیستم به صورت دیاگرام‌های ویوچارت مدل می‌شود و مدل ایجاد شده به دو گروه تولیدکنندگان کد و موارد تست تحویل داده می‌شود. این دو گروه به صورت موازی مشغول به کار می‌شوند. با پیشرفت تولید کد، موارد تست متناسب تولید می‌شوند و این بخش‌ها تست می‌شوند. این ویژگی از جزئی(جزء به جزء) بودن نمودارهای ویوچارت سرچشمه می‌گیرد. به این ترتیب سیستم به صورت پی در پی توسط موارد تست کوچک تست می‌شود.

به دلیل تولید موارد تست جزئی این روش نسبت به روش‌های دیگر تولید موارد تست سرعت بیشتری دارد و برای تست سیستم‌هایی که زمان کمتری برای تولید موارد تست دارند، بسیار مناسب است. از طرفی با توجه به ساختار مبتنی به دیدگاه آن، برای تست سیستم‌های چند کاربره مانند سیستم‌های تحت شبکه یا تحت وب کارایی

---

[1] Partial
[2] Prototype
[3] Use Interaction Test

بالایی دارد؛ از طرفی به دلیل قوانین کنترل حوزه موجود در آن حتی سطوح دسترسی کاربران و امنیت سیستم را نیز می‌توان تا حدی تست کرد که کاربرد بسیار زیادی خواهد داشت و متدولوژیهای خاص در انتخاب موارد تست که مبتنی برکد برنامه مورد تست بودند پاسخ داده می‌شد[5].

اما استفاده از کد برنامه برای تولید موارد تست باعث می شد تا طراحان برنامه مجبور باشند برای تولید موارد تست تا کامل شدن کد برنامه منتظر باشند و از طرفی استخراج موارد تست از کد برنامه فعالیتی وقت گیر و پرهزینه است و در ضمن پیچیدگی زیادی دارد و در کل استفاده از کد برنامه برای تولید موارد تست سبب طولانی شدن دوره ساخت و تحویل نرم افزار می‌شود.

به دلیل این ضعف ها در این روش، تولید کنندگان تست به دنبال منابع دیگری برای تولید موارد تست رفتند. این منابع، ابزارهای پیش از کد نظیر نیازمندی ها، مشخصات[1] و مدل‌های طراحی بودند. استفاده از این گونه ابزارها سبب می شد تا طراحان تست مجبور نباشند تا پایان کار توسعه دهندگان کد برنامه صبر کنند و بعد از کامل شدن مدل برنامه، کار خود را به موازات کار توسعه دهندگان کد برنامه آغاز کنند.

[1] Specifications