پایان نامه کارشناسی ارشد

روشی خودکار برای آزمون مبتنی بر مدل برنامه‌های تحت وب

تهیه و تنظیم: حمیده حاجی آبادی

استاد راهنما: دکتر محسن کاهانی

سال ۸۸

فهرست مطالب

فهرست جداول

فهرست اشکال

فصل ۱. مقدمه

۱–۱ آزمون نرم افزار

آزمون نرم افزار فرایندی است که کیفیت نرم افزار کامپیوتری را مشخص میکند. آزمون نرم افزار شامل فرایند اجرای یک برنامه با هدف یافتن باگهای نرم افزاری است، اما محدود به آن نمی‌باشد. کیفیت مطلق نیست، بلکه برای افراد مختلف نسبی است. با این تصور، آزمون کردن هرگز نمیتواند صحت نرم افزارهای کامپیوتری دلخواه را به طور کامل اثبات کند. یک نکته مهم اینست که آزمون نرم افزار، باید از نقطه نظرات مختلفی از تضمین کیفیت نرم افزار لحاظ شود، که با همه حوزه های فرایند تجاری همراه باشد نه فقط حوزه های آزمون.

در بازه حضور نرم افزارهای کامپیوتری، پیچیدگی و اندازه آنها همواره رو به افزایش است. هر محصول نرم افزاری مخاطبان خاصی دارد. برای مثال یک نرم افزار بازی کامپیوتری مخاطبانی کاملا متفاوت از مخاطبان یک نرم افزار بانکی دارد. بنابراین،زمانیکه سازمانی یک محصول نرم افزاری را مینویسد یا خریداری میکند ، منطقا باید اطمینان حاصل کند که آیا محصول نرم افزاری برای کاربران، مخاطبان، خریدارآن و سایر سهامدارانش قابل پذیرش هست یا نه. آزمون نرم افزار فرایند تلاش برای این گونه ارزیابی هاست.

طی مطالعه ای که توسط NIST[1] در سال ۲۰۰۲ انجام شد، باگهای نرم افـزاری بـرای اقتصـاد آمریکـا سالانه ۵۹,۵ بیلیون دلار هزینه دارد. اگر فرایند آزمون نرم افزار بهتر انجام شود، بیشتر از یـک سـوم ایـن هزینه قابل اجتناب است.

آزمون نرم افزار ممکن است به عنوان یک قسمت مهم از فرایند تضمین کیفیت نرم افزار تلقـی شـود. در تضمین کیفیت نرم افزار متخصصان فرایند نرم افزار دیدگاه وسیعتری روی نرم افزار و توسعه آن دارند. آنها فرایند مهندسی نرم افزار را بررسی میکنند و آنرا برای کاهش میزان خطاهای منجر به شکست، تغییر میدهند. عنصر تعیین کننده در میزان شکست قابل قبول، ماهیت نرم افزار است. بازی ویدئویی کـه بـرای شبیه سازی پرواز یک هواپیما طراحی شده، منطقاً باید نسبت به نرم افزاری که برای کنترل یک خط پرواز واقعی به کار میرود، تحمل شکست بیشتری داشته باشد.

شکست نرم افزار از طریق فرایندهای زیر رخ میدهد. برنامه نویس خطایی را انجام میدهد که منجر به یک شکست در کد منبع نرم افزار میشود. اگر این خطا کامپایل و اجـرا شـود، در موقعیـت هـای خاصـی سیستم نتایج نادرستی تولید میکند که منجر به شکست میشود. لزومـاً همـه خطاهـا منجـر بـه شکسـت نمیشوند. برای مثال خطا در کدهایی از برنامـه کـه هرگـز اجـرا نمـی‌شـوند منجـر بـه شکسـت نخواهـد شد.[Kan99]

یک مسئله در آزمون نرم افزار حتی زمانیکه فقط یک محصول ساده را آزمـون میکنـیم، اینسـت کـه آزمون همه ترکیبات ممکن ورودی و پیش شرایط ممکن نیست. این یعنی تعداد خطاها در یک محصـول نرم‌افزاری میتواند خیلی زیاد باشد. خطا هایی که بصـورت نامتنـاوب رخ میدهنـد، بـه سـختی در فراینـد آزمون قابل ردیابی می‌باشند. به بیان علمی می‌توان گفت،کیفیت کاملاً وابسته به نظر افراد است و ممکـن است کیفیتی که به نظر فردی مناسب است از لحاظ فردی دیگر غیر قابل قبول باشد.

[1] http://www.nist.gov/public_affairs/releases/n02-10.htm

روشهای مختلفی برای آزمون نرم افزار وجود دارد. دوباره بررسی کردن و امتحانات سخت و مهم، آزمونهای ایستا نامیده می‌شود، در حالیکه اجرای واقعی برنامه با یک مجموعه آزمون داده شده در مرحله خاصی از فرایند توسعه، آزمون پویا نامیده میشود.

آزمون نرم افزار در ارتباط با تضمین اینکه آیا نرم افزار را به درستی تولید کرده ایم (verification) و نیز تضمین اینکه آیا نرم افزار درست کار میکند (validation)، به کار میرود.

آزمون نرم افزار توسط آزمون کنندگان نرم افزار انجام میشود. تا قبل از سال ۱۹۵۰ عبارت آزمون کننده نرم افزار به صورت کلی به کار میرفت، اما بعدها به عنوان یک حرفه مجزا مطرح شد. با توجه به بازه‌ها و اهداف متفاوت در آزمون نرم افزار نقشهای متفاوتی در این حوزه عنوان شده اند، از جمله، مدیر یا راهبر آزمون، آزمون کننده، طراح آزمون، خودکار کننده آزمون یا طراح اتوماسیون و مدیر آزمون.

۲-۱ آزمون مبتنی بر مدل

مفهوم آزمون مبتنی بر مدل به اوایل دهه هفتاد برمی‌گردد یعنی زمانی که آزمون بر مبنای مشخصات منسوخ شد. مقاله هایی در سال ۲۰۰۰ درباره رشد علاقه به آزمون مبتنی بر مدل در زمینه های تجاری و آکادمی منتشر شد. گزارشات این تجربیات نشان میدهد که آزمون مبتنی بر مدل برای برنامه های کوچک مثل سیستمهای تعبیه شده[1] و رابط کاربر خوب کار می کند. تحقیقات بر روی اینکه آیا آزمون مبتنی بر مدل برای سیستمهای بزرگ مثل برنامه های کاربردی وب مناسب است یا خیر هنوز ادامه دارد.

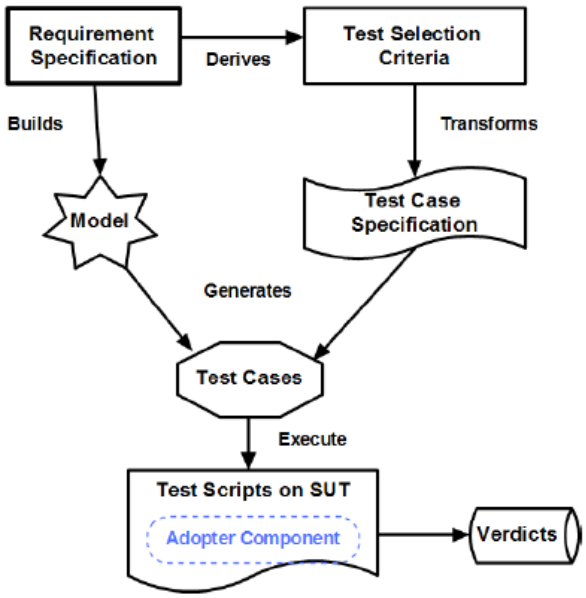
مقاله های زیادی وجود دارد که بر آزمون مبتنی بر مدل متمرکز شده اند و آزمون مبتنی بر مدل را از جنبه‌های مختلفی مورد بررسی قرار داده اند. برخی از این مقالات بر فرایند آزمون مبتنی بر مدل و بعضی

[1] embedded

دیگر بر روی متدهایی برای پشتیبانی آزمون مبتنی بر مدل و برخی دیگر دامنه برنامه های کاربردی را برای آزمون مبتنی بر مدل مورد بررسی قرار می‌دهند.

دایره المعارف WWW می‌گوید: آزمون مبتنی بر مدل، آزمون یک نرم‌افزار است که نمونه‌های آزمون از جزء یا کل یک مدل مشتق شود. مدل برخی (تمام) جنبه‌های سیستم زیر آزمون را به بیان ساده‌تر نشان می‌دهد. SUT می‌تواند چیزی به سادگی یک متد یا کلاس، یا به پیچیدگی یک سیستم یا چندین سیستم باشد. برای انجام عمل آزمون، مدل که رفتار سیستم را مشخص می‌کند، برای تولید مجموعه‌ای از نمونه‌های آزمون پردازش می‌شود. برای پاسخ به اینکه آیا SUT با ویژگیهای خواسته شده‌ای که در مدل نشان داده شده است، مطابقت دارد، نمونه‌های آزمون بر روی نرم‌افزار اعمال می‌شوند. فرایند کلی آزمون مبتنی بر مدل شامل ایجاد مدل، تعریف معیارهای انتخاب آزمون، تبدیل آن به مشخصه‌ها، ایجاد آزمونها، نصب و اجرای نمونه‌های آزمون بر روی SUT می‌باشد. شکل زیر فرایند کلی آزمون مبتنی بر مدل را نشان می‌دهد.

مراحل فرایند آزمون مبتنی بر مدل بصورت شکل ۱-۱ تعریف می‌شوند:



شکل ۱-۱ فرایند آزمون مبتنی بر مدل

ایجاد مدل: مدل برمبنای مشخصه‌هایی که رفتار مورد انتظار SUT را نشان میدهد، ایجاد می‌شود.

تعریف معیار انتخاب آزمون: معیار انتخاب آزمون مناسب قابلیت انتخاب نمونه آزمون "خوب" را برای SUT بیان می‌کند. در تئوری یک نمونه آزمون خوب، آزمونی است که خطاهای احتمالی را با هزینه قابل قبول پیدا می‌کند. بهرحال در حالت کلی تعریف نمونه آزمون خوب کار ساده‌ای نیست.

تبدیل به مشخصه‌ها: مشخصه‌های نمونه آزمون مفهوم معیار انتخاب آزمون را روشن می‌کند و آنها را عملی می‌کند. در بعضی نمونه‌ها مولد آزمون در صورتی که مدلها و مشخصه‌ها را داشته باشد، می‌تواند بطور خودکار رشته آزمونها را تولید کند.

ایجاد آزمونها: نمونه آزمونهای زیادی در دسترس است، مولد آزمون بطور تصادفی تعدادی از این نمونه‌ها را از بین رنج وسیع نمونه برمی‌دارد.

اجرای آزمونها: در این مرحله رشته آزمونهای انتخاب شده ایجاد شده‌اند و نمونه آزمونها در حال اجرایند، چون مدلها و SUT در دو سطح انتزاعی متفاوتند، لازم است پلی بین آنها قرار گیرد. که با نصب اجزای تطبیق دهنده برای تبدیل اجزای ورودی نمونه‌های آزمون به داده‌های قابل قبول برای SUT و انتزاع قسمتهای خروجی انجام می‌شود. Verdict نتیجه مقایسه خروجی SUT با خروجی مورد انتظار است، که میتواند قبول یا رد باشد. [Mak07]

۳–۱ مبانی آزمون نرم افزار

آزمون برای مهندس نرم افزار امری غیر عادی تلقی می شود. طی مراحل اولیه تعریف وتوسعه، مهندس می کوشد تا نرم افزار را از یک مفهوم انتزاعی، به یک محصول عینی تبدیل کند. اکنون نوبت به آزمون می رسد. در این مرحله وی به منظور تخریب نرم افزار ساخته شده به طراحی نمونه‌های آزمون می‌پردازد. در حقیقیت، آزمون، تنها مرحله ای از فرآیند نرم افزار است که بیشتر ویرانگر (حداقل از نظر

روآن شناختی) به نظر می رسد تا سازنده. مهندسآن نرم افزار، فطرتا" افرادی سازنده هستند. آزمون، مستلزم آن است که سازنده دید پیش داوری خود، مبنی بر درستی نرم افزار را دور بریزد. (Li, 2007)

۴-۱ هدف آزمون

"گلن مایزر" درباره نرم افزار چند قاعده را بیان کرد که به خوبی به اهداف آزمون را بیان می‌کند.

آزمون فرآیند اجرای برنامه به قصد یافتن خطاهاست.

نمونه آزمون[1]خوب، نمونه‌ای است که احتمال یافتن خطاهای کشف شده در آن، بالا باشد.

آزمون موفق، آزمونی است که خطاهای کشف نشده را کشف می کند.

اهداف بالا نشانگر یک تغییر دیدگاه زیبا است، و برخلاف این دیدگاه عامیانه که آزمون موفق، آزمونی است که در آن خطایی یافت نشود. اگر آزمون با موفقیت اجرا شود (مطابق اهداف مذکور)، خطاهای نرم افزار را بر ملا خواهد نمود. به عنوان مزیت دوم، آزمون نشان می دهد که رفتار نرم افزار ظاهرا" مطابق مشخصه است و خواسته های رفتاری و کارایی ظاهرا" برآورده شده اند. به علاوه، داده های جمع آوری شده به موازات انجام آزمون، شاخص خوبی از قابلیت اطمینآن و کیفیت نرم افزار به دست می دهند. ولی آزمون نمی تواند نبود خطاها و نقایص را ثابت کند. بلکه فقط می تواند وجود خطا و نقایص را اثبات کند. [YeL07]

۵-۱ اصول آزمون

[1] Test case

مهندس نرم افزار پیش از اعمال روش ها در خصوص نمونه‌های آزمون مؤثر، باید اصول پایه‌ای را که آزمون نرم افزار را هدایت می کند، درک کند. "دیویس" مجموعه ای از اصول پیشنهاد می کند، که در اینجا از آنها استفاده خواهیم کرد:

همه آزمون ها باید تا حد خواسته های مشتری قابل ردیابی باشند. همانطور که قبلا اشاره شد، هدف آزمون نرم افزار کشف خطاها است. بدین معنی که اکثر نقایص بزرگ (از دیدگاه مشتری) آنهایی هستند که باعث می شوند برنامه نتواند خواسته‌هایشان را برآورده کند.

باید مدت ها قبل از شروع آزمون، طرح ریزی شود. طرح ریزی آزمون می تواند به محض کامل شدن مدل نیازمندی‌ها شروع شود. تعریف مشروح نمونه‌های آزمون می تواند به محض منسجم شدن مدل طراحی آغاز شود. بنابراین، همه آزمون ها را می توان پیش از تولید هر گونه کد، برنامه ریزی و طراحی کرد.

اصل "پارتو" در آزمون نرم افزار نیز صدق می کند. به عبارت ساده، اصل "پارتو" بیان می کند که ۸۰ درصد همه خطاهای کشف شده، احتمالا" در ۲۰ درصد کد برنامه وجود دارد. مسئله، جدا سازی مؤلفه های مظنون و آزمودن کامل آنهاست.

آزمون باید در ابعاد کوچک آغاز شود و به ابعاد بزرگتر گسترش یابد. اولین آزمون ها برروی هر یک از مؤلفه ها انجام می شوند. با پیشرفت آزمون، خطاهای مجموعه ای از مؤلفه های مجتمع و سپس کل سیستم یافت می شود.

آزمون کامل امکان پذیر نیست. تعداد مسیرهای ممکن برای یک برنامه با اندازه متوسط نسبتا زیاد است. لذا، اجرای هر ترکیبی از مسیرها امکان پذیر نیست. ولی این امکان وجود دارد که برنامه را در حد کفایت پوشش دهیم.

برای آنکه آزمون بیشترین بازدهی را داشته باشد، باید توسط یک شخص ثالث و بی طرف انجام شود. منظور از بیشترین بازدهی آن است که خطاها را با احتمال بیشتری بیابد. به دلایلی کـه قـبلا" در همـین فصل ذکر شد، مهندس نرم افزاری که سیستم را ایجاد کرده است، بهترین کسـی نیسـت کـه بایـد همـه آزمون ها را انجام دهد. [YeL07]

۶–۱ آزمون پذیری نرم افزار ها

در شرایط ایده آل، مهندس نرم افزار یک برنامه کامپیوتری، یک سیستم یـا یـک محصـول را طـوری طراحی می کند که آزمون پذیری داشته باشد. به ایـن ترتیـب افـرادی کـه آزمـون را انجـام مـی دهنـد، می‌توانند نمونه‌های آزمون کارآمد را آسانتر طراحی کنند.

آزمون پذیری چیست؟ "جیمزبک"، آزمون پذیری را به شیوه زیر تعریف می کنـد: آزمـون پـذیری نرم افزار میزانی از سهولت آزمودن یک برنامه کامپیوتری است. چـون آزمـودن دشـوار اسـت، بهتـر اسـت بدانیم چه چیزی آن را به جریان می اندازد. گاهی برنامه نویسان مشتاق به انجام کارهایی هستند کـه بـه فرآیند آزمون کمک می کند و لیست کنترلی از نکات طراحی، ویژگی ها و غیره می تواند در بحث با آنهـا مفید واقع شود.

معیارهایی وجود دارد که جنبه های مختلف آزمون پذیری را اندازه گیـری مـی‌کنـد، گـاهی، آزمـون پذیری به این معنا است که چگونه مجموعه خاصی از آزمون ها، محصول را تحت پوشش قرار می دهد. در ارتش به معنای این است که یک ابزار چگونه در میدان جنگ قابل کنتـرل وتـرمیم اسـت. ایـن دو معنـا، "قابلیت آزمون نرم افزار" را توصیف نمی کنند. لیست کنترلی که در ادامه می آید، ویژگی هایی را مطرح می کند که نرم افزار آزمون پذیر را ایجاد می نماید:

قابلیت کار[1] "هر چه بهتر کار کند، آزمون آن کارآمدتر است".

قابلیت مشاهده[2] "آنچه می بینید، همان است که آزمایش می کنید".

کنترل پذیری[3] "هر چه بهتر بتوان نرم افزار را کنترل کرد، آزمون را بیشتر می‌توان خودکار و بهینه کرد".

تجزیه پذیری[4] "با کنترل دامنه کاربرد آزمون، می توان مسائل را سریعتر جدا سازی کرد، و آزمون های مجدد را با هوشمندی بیشتر انجام داد".

سادگی[5] "هر چه مورد آزمون کوچکتر باشد، سریعتر می توان آن را آزمود".

پایداری[6] "هر چه تعداد تغییرات کمتر باشد، آزمون کمتر با مانع مواجه می شود".

درک *پذیری*[7] "هر چه اطلاعات بیشتری داشته باشیم، آزمون هوشمندانه تر انجام می شود".

مهندس نرم افزار می تواند با استفاده از صفاتی که "بَک" توصیه می کند، پیکر بندی نرم افزار (یعنی برنامه ها، داده ها و مستندات) را طوری توسعه دهد که مستعد آزمون باشد.

۷-۱ سطوح مختلف آزمون

آزمون نرم افزار در فازهای متفاوت و در سطوح مختلف انجام می پذیرد. این سطوح عبارتند از:

---

[1] Operability

[2] Observability

[3] Controllability

[4] Decomposability

[5] Simplicity

[6] Stability

[7] Understandability

۱–۷–۱ آزمون واحد

یکی از مراحل اولیه آزمون یک سیستم، آزمون واحد[1] می باشد که هر یک از واحدها یا ماژولهای تشکیل دهنده یک برنامه را بطور مستقل، مورد آزمون قرار می دهد. معمولا آزمون واحد توسط خود برنامه نویسان و به موازات توسعه سیستم انجام می شود. یعنی هر برنامه نویسی که ماژولی را برای سیستم می‌نویسد، وظیفه آزمون آن را نیز بعهده دارد و در همان زمان قابل انجام است.

هدف از انجام آزمون واحد، اطمینان از درستی عملکرد واحدهایی است که پس از توسعه، در قسمتهای مختلف سیستم مورد استفاده قرار خواهند گرفت.

آزمون واحد، معمولا جزء آزمونهای جعبه سفید به حساب آورده می شود که نیاز است به ساختار درونی کد مورد آزمون دسترسی داشته باشد.

لازم به ذکر است که علیرغم اهمیت بسیار زیاد آزمون واحد در فرآیند تضمین کیفیت نرم افزار، این نوع آزمون نمی تواند جایگزین دیگر انواع آزمون شود. بعنوان مثال، با استفاده از آزمون واحد نمی توان کیفیت رابط گرافیکی سیستم را ارزیابی نمود یا آزمون بار را نمی توان با استفاده از آزمون واحد انجام داد[Ham04].

۲–۷–۱ آزمون یکپارچگی

هدف از آزمون یکپارچگی سیستم[2] آن است که مطمئن شویم اجزای مختلف سیستم، در کنار یکدیگر، بخوبی کار می کنند و تعاملات، ارتباطات و رد و بدل کردن داده ها در بین ماژولهای مختلف سیستم، بدرستی انجام می شود و در نتیجه، کل سیستم عملکرد صحیحی دارد.

---

[1] Unit testing

[2] Integrity testing

آزمون یکپارچگی را می توان در سطوح متفاوتی انجام داد. مثلا، می توان هر یک از ماژولهای اساسی سیستم را بعنوان یک سیستم در نظر گرفت (که خودش از اجزای کوچکتری تشکیل شده) و آزمون یکپارچگی را در مورد آن انجام داد. همچنین می توان کل سیستم را بعنوان یک سیستم واحد در نظر گرفته و آن را مورد آزمون قرار داد.

نکته قابل توجه آن است که نباید اینطور تصور شود که انجام آزمون واحد بر روی ماژولهای سیستم، ما را از انجام آزمون یکپارچگی بی نیاز می کند. در واقع هر دو نوع آزمون مذکور لازم می باشند و هر یک توانایی خاص خود را دارند. نقطه مورد توجه آزمون یکپارچگی، نقاط تماس و تعامل ماژولها با یکدیگر است و ماژولها را در کنار یکدیگر و در ضمن کار با هم، مورد آزمون قرار می دهد، در حالیکه آزمون واحد، ماژولها را بطور مستقل و جدا از بقیه اجزای سیستم مدنظر قرار می دهد. [Cra02] [Chu05]

۳-۷-۱ آزمون سیستم

یک سیستم کاملاً یکپارچه را آزمون می کند تا بررسی کند که آیا تمام نیازمندیها براورده می شوند یا نه. قبل از عرضه نسخه نهایی یک نرم افزار ، آزمونهای آلفا و بتا نیز علاوه بر آزمونهای فوق انجام می‌شوند.

آزمون الفا معمولاً برای نرم افزار های تولید انبوه به عنوان نوعی از آزمون پذیرش بکار برده می شود. و قبل از مرحله آزمون بتا صورت می پذیرد .

نسخه هایی از نرم افزار ، که نسخه‌های بتا نامیده میشوند، به مخاطبان محدودی در خارج از تیم برنامه نویسان عرضه می شود. هدف اینست تا آزمونهای بیشتری توسط افراد دیگری روی نرم‌افزار انجام شود و تا اطمینان حاصل شود که نرم افزار خطا یا باگهای کمی دارد. کاهی اوقات نسخه های بتا به عموم عرضه می‌شود تا میزان بازخورد ها افزایش یابد .

۴-۷-۱ آزمون پذیرش[1]

این نوع آزمون بر اساس نیازمندیهای مستند شده کاربران سیستم انجام می شود و هدف از انجام آن کسب اطمینان از تامین نیازهای کاربران توسط سیستم، می باشد. به بیان دیگر در این نوع آزمون می‌خواهیم مطمئن شویم که سیستم تولید شده از دید کاربران قابل قبول است یا خیر. بهمین علت بهتر است انجام آزمون توسط خود کاربران یا نمایندگان آنها در محیط و شرایط واقعی صورت گیرد. [Chu05] [Cra02]

در نهایت، آزمون پذیرش توسط کاربران نهایی یا مشتریان انجام میشود تا پذیرش محصول صورت گیرد. آزمون پذیرش ممکن است به عنوان بخشی از فرایند، زمانیکه از یک فاز توسعه به فاز دیگر میرویم نیز صورت گیرد.

[1] acceptance testing

فصل ۲.روشهای موجود آزمون مبتنی بر مدل برای سیستم های مبتنی بروب

۲–۱ مقدمه

برنامه‌های کاربردی تحت وب روز به روز پیچیده تر می‌شوند بطوریکه امروزه در رشته نرم‌افزار تبدیل به یکی از زمینه‌هایی شده است که بیشترین رشد را دارد. نه تنها هر روز برنامه تحت وب جدیدی ایجاد می‌شود بلکه روز به روز روشهای توسعه جدیدی نیز تولید می‌شوند. برنامه‌های تحت وب غالبا برنامه‌هایی با تعامل بسیار زیادند که نیاز به کیفیت بالایی دارند. برای اطمینان از کیفیت آنها نیاز به استفاده از روشهای سیستماتیک برای آزمون می‌باشد. یکی از این روشها استفاده از آزمون مبتنی بر مدلست. در این روش مدلها نقش کلیدی دارند، مدل برنامه کاربردی برای توضیح سیستم در حال آزمون ایجاد می‌شود، مدل نمونه‌های آزمون از روی مدل سیستم ایجاد می‌شود، سپس موتور آزمون نمونه‌های آزمون را اجرا کرده و نتیجه آزمون را به نمونه آزمون بر می‌گرداند.

اما برنامه‌های تحت وب غالبا بدون دنبال کردن مدل فرایند رسمی توسعه می‌یابند: نیازمندیها ضبط نمی‌شوند، فاز طراحی در نظر گرفته نمی‌شود و هیچ سند رسمی‌ای تولید نمی‌شود. بعبارتی توسعه دهندگان خیلی زود وارد فاز پیاده‌سازی می‌شوند. چالش امروز برنامه وب همین است. توسعه دهنده

برنامه وب، مدلی برای توسعه ندارد و از همان ابتدا شروع به کد نویسی می‌کند. یا حتی اگر در ابتدا مدلی طراحی شده باشد معمولا در فرایند انجام کار به روز نمی‌شود، و در نتیجه مدل ضعیفی است، و شاید با سیستم پیاده سازی شده کاملا متفاوت باشد. پس برای آزمون برنامه‌های تحت وب نمی‌توان امیدوار به داشتن چنین مدلی بود. بنابراین باید به شیوه‌ای مدل را استخراج کرد.

در ادامه روشهای آزمون مبتنی بر مدل برای سیستمهای مبتنی بر وب، بررسی می‌شود، روشهایی بررسی می‌شوند که ابتدا مدل سیستم را استخراج می‌کنند و سپس از روی مدل استخراج شده فرایند آزمون صورت می‌پذیرد.

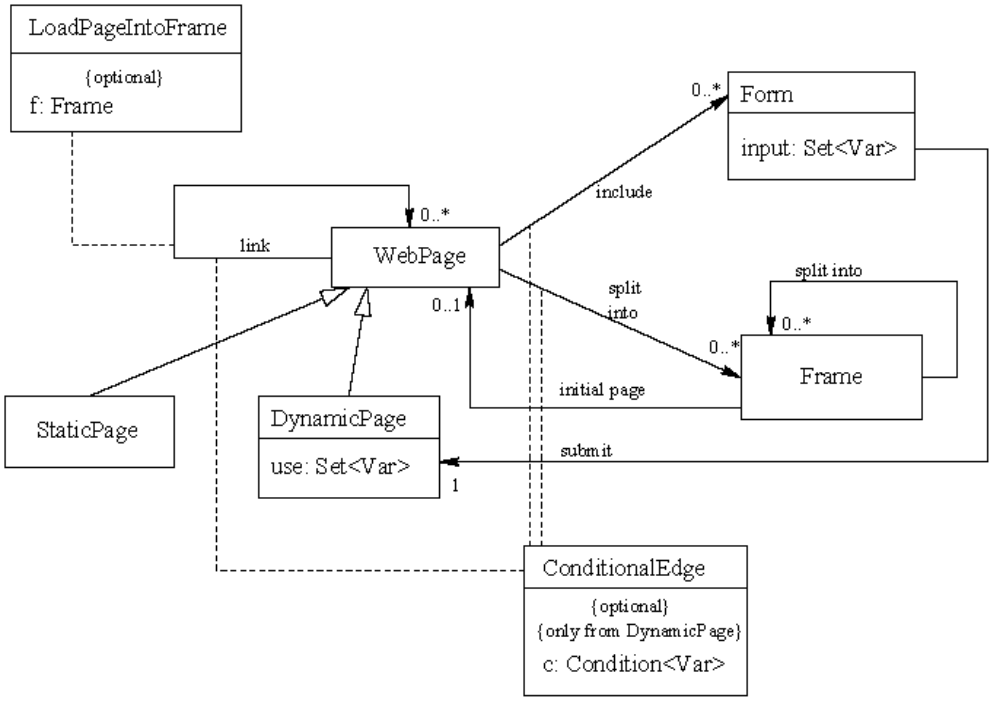
۲-۲ ابزارهای موجود

در اینجا ابزارهای موجود در زمینه آزمون برنامه‌های کاربردی تحت وب با استفاده از آزمون مبتنی بر مدل بررسی می‌شوند. ابزارهایی مورد بررسی قرار می‌گیرد، که ابتدا مدل ساختاری سیستم را استخراج و سپس به آزمون آن می‌پردازند.

۱-۲-۲ ابزارهای ReWeb and TestWeb

Filippo Ricca و Paolo Tonella در سال ۲۰۰۱ دو ابزار برای مهندسی مجدد و آزمون برنامه‌های تحت وب ارائه دادند. آنها متا مدلی برای برنامه‌های وب پیشنهاد دادند. سپس ابزاری به نام reWeb توسعه دادند، که آدرس و کد منبع سایت را به عنوان ورودی می‌گرفت و ساختاری مطابق با متا مدل ارئه شده استخراج می‌کرد. آنها کارشان را در سالهای بعد کاملتر کردند و ابزاری مبتنی بر مدل به نام TestWeb توسعه دادند. testWeb مدل استخراج شده توسط reWeb را بعنوان ورودی می‌گرفت و براساس آن

نمونه‌های آزمون را ایجاد می‌کرد. ساختار متا مدلی که آنها ارائه کردند در شکل ۱ نمایش داده می‌شود. [Ric01] [Ric04] [Ric01,2]

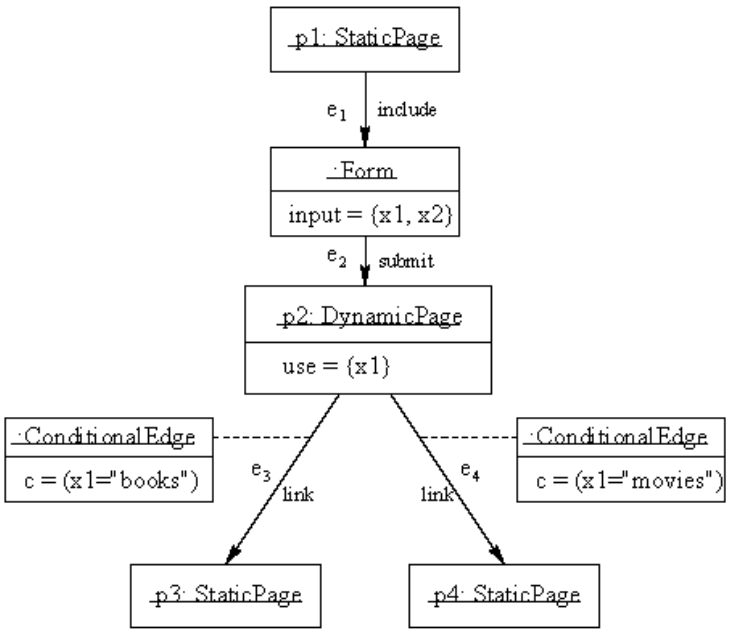


شکل ۲-۱ متا مدل ارائه شده توسط F. Ricca و P. Tonella

همانطور که در شکل ۲-۱ مشخص است کلاس WebPage شامل تعدادی فرم و فریم می‌باشد، که به دو دسته صفحات استاتیک و صفحات دینامیک تقسیم می‌شود. صفحه استاتیک صفحه ای است که محتوایش ثابت و در محلی پایا ذخیره شده است اما صفحه دینامیک از سمت سرور محتوایش تغییر می‌کند و غالبا وابسته به ورودیهاییست که توسط فرمها با دو متد post یا get به سرور ارسال می‌شود. متغیرهای فرم بصورت use در کلاس dynamic page تعریف شده اند. وقتی یک لینک در web page باعث بارگذاری صفحه در فریم دیگر می‌شود، آن فریم داده‌ای از کلاس LoadPageIntoFram می‌باشد. همانطور که در شکل ۲-۱ مشخص است، در ابتدا در هر فریم یک صفحه بارگذاری می‌شود. (لینک initial page) لینک submit باعث فرستادن داده‌های فرم به سرور می‌شود، که در پاسخ به آن یک

dynamic page ایجاد می‌شود. conditionalEdge کلاسی است که به هر association منسوب می‌شود و اگر شرط برقرار شود صفحه لینک شده بارگذاری می‌شود.

بعنوان مثال مدل یک وب سایت درشکل ۲-۲ ارائه شده است، این مدل نمونه‌ای از متا مدل فوق می‌باشد.



شکل ۲-۲- متا مدل یک وب سایت نمونه

در مدل شکل ۲-۲، در صفحه p1 امکان جستجو با یک فرم ایجاد می‌شود، و صفحه دینامیک p2 نتیجه جستجو را نشان می‌دهد. اگر نتیجه جستجو book یا movie باشد، لینک به صفحه دیگر برای نمایش اطلاعات اضافی فعال می‌شود.

پس از فاز مدلسازی وارد فاز آزمون می‌شوند. Ricca در مقاله اش به دو فاز آزمون استاتیک و آزمون دینامیک اشاره کرده است.

فاز آزمون استاتیک: در این فاز از انجام آزمون خطاهای مربوط به صفحات غیر قابل حصول، صفحات سرگردان و وابستگی داده‌ها آشکار می‌شود.

فاز آزمون دینامیک: برای انجام این نوع آزمون از آزمون جعبه سفید استفاده شده است، بدین معنی که کد منبع برنامه به منظور استخراج ساختار مورد نیاز پویش و آنالیز می‌شود.

نمونه آزمون برای سیستمهای مبتنی بر وب شامل مجموعه‌ای از URLها و ورودی هاست. بنابراین برای تولید یک نمونه آزمون باید رشته URLها و ورودیها تولید شود. در ابزار مذکور رشته URLها بصورت خودکار و با برخی الگوریتمهای پیمایش مدل (گراف) تولید می‌شوند، و ورودیهای مورد نیاز بصورت دستی به نمونه آزمون اضافه می‌شود.

پس از اجرای نمونه آزمون verdict بصورت دستی به نمونه آزمون اضافه می‌شود. بدین معنی که پس از اتمام اجرای هر نمونه آزمون، کاربر صفحه را باز کرده و نتیجه قبول یا رد را بصورت دستی به نمونه آزمون اضافه می‌کند.

۲-۲-۲ روش مبتنی بر جلسه[1]

آقای Elbaum در سال **۲۰۰۳** با استفاده از sessionها روشی را برای آزمون برنامه‌های وب ارائه داد، همچنین روشی را برای بهبود روش Ricca پیشنهاد داد، که در ذیل به توضیح آن می‌پردازیم.

در روشی که آقای Ricca ارائه کرده بود، برای تولید نمونه آزمون رشته URLها بصورت خودکار از پیمایش گراف ایجاد می‌شد و ورودیها بصورت دستی به نمونه آزمون اضافه می‌شدند. آقای Elbaum روشی را برای تولید ورودیها ارائه داد که به خودکار کردن روش Ricca کمک می‌کرد. بدین ترتیب که بعد از تولید رشته URLها، برای تولید ورودیها در فایل log جستجو می‌شود که در کدام session دقیقا همان URLsequence وجود دارد، سپس inputهای متناظر از فایل log برداشته می‌شوند و بعنوان ورودی‌های

[1] Session based

آزمون از آنها استفاده می‌شود. بعبارتی دیگر به جای اضافه کردن دستی ورودیها به نمونه آزمون، در فایل log جستجو و داده‌ها از آنجا برداشته می‌شود. [Elb03]

۳-۲-۲ ابزار FSMWeb

Anneliese A. Andrews در سال ۲۰۰۵ یک ابزار جعبه سیاه به نام FsmWeb به منظور آزمون مبتنی بر مدل برای سیستمهای مبتنی بر وب ارائه داد که نمونه‌های آزمون را بر اساس مدل FSM سیستم تولید می‌کرد. آقای Andrews مشکل انفجار حالت[1] مدلهای FSM[2] را با جمع کردن FSM[3]های سلسله مراتبی حل کرد. بدین ترتیب که برای آزمون در سطح سیستم، نمونه‌های آزمون سطوح پایینتر (نمونه‌های آزمون FSMهای لایه زیر) با هم الحاق می‌شوند. [And05]

او برای تولید FSMهای سلسله مراتبی نرم‌افزار زیر تست را به چندین کلاستر تقسیم کرد، عمل کلاستر بندی را تا سطحی ادامه داد که نودها در FSM مربوط به کلاستر مورد نظر صفحات وب باشند. سپس برای تولید مدل سیستم ابتدا از سطح پایین شروع می‌کند و FSMهایی که نودهای آنها صفحات وب هستند، ایجاد می‌شوند. سپس سطوح بالاتر که در آنها هر نود یک FSM است، تولید می‌شوند. اگر یک صفحه از کلاستر a به یک صفحه در کلاستر b لینکی داشته باشد، آنگاه در مدلهای سطح بالاتر یک لینک از کلاستر a به کلاستر b وصل می‌شود. هر کلاستر فقط یک نود خروجی دارد، و اگر چنین نودی موجود نباشد، بصورت مجازی ایجاد می‌شود.

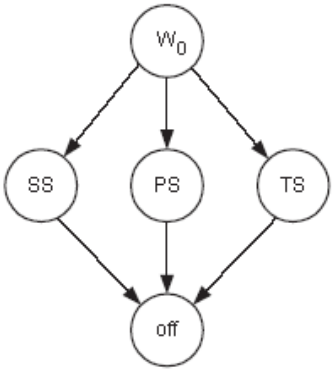
در شکل‌های ۲-۳، ۲-۴ و۲-۵ سه سطح مدلسازی برای یک سایت نشان داده شده است:

---

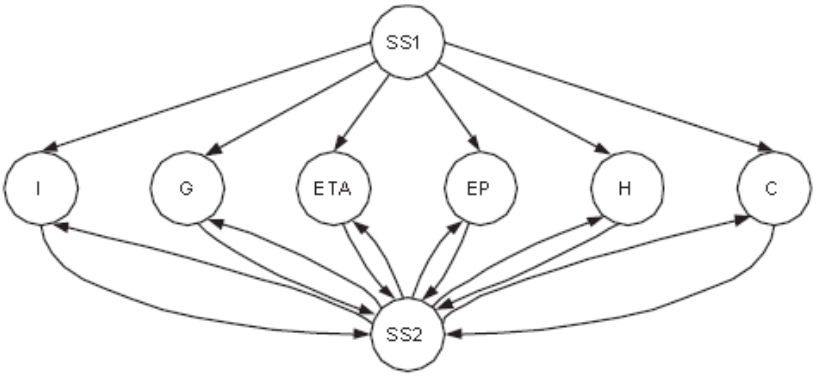
[1] State explosion

[2] Finite State Machine

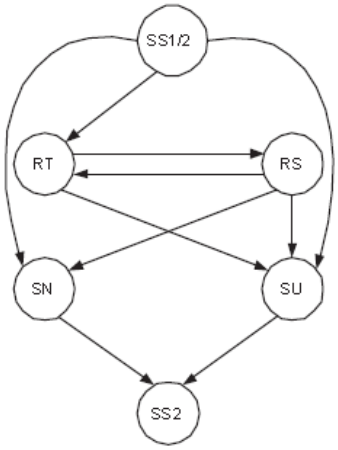
[3] aggregate



شکل ۲-۳- مدل سطح بالا. SS, PS, TS سه کلاستر هستند



شکل ۲-۴- مدل کلاستر SS



شکل ۲-۵- مدل سطح ۳ برای نود I

لبه‌ها، هم در سطح logical web pageها و هم در سطح FSMها با ورودیها و محدودیتها یادداشت نویسی[1] می‌شوند. ابتدا نمونه‌های آزمون پایینترین سطح تولید می‌شوند. برای تولید نمونه‌های آزمون سطوح بالاتر، نمونه‌های آزمون سطوح پایینتر، با هم الحاق می‌شوند. یعنی رشته آزمونهای ماشین حالت متناهی f با رشته آزمونهای ماشین حالت متناهی g الحاق می‌شوند و Annotate A در میان آنها قرار می‌گیرد. اگر ماشین f تعداد n رشته آزمون و ماشین g تعداد m رشته آزمون داشته باشد، آنگاه در سطح بالاتر پس از جمع کردن دو ماشین فوق m*n رشته آزمون تولید می‌شود.

ابزار FsmWeb ۴ فاز زیر را برای تولید نمونه‌های آزمون دنبال می‌کند:

- ایجاد مدل سیستم
- تعریف معیار انتخاب برای داده‌های ورودی و برای رشته URLها
- ایجاد نمونه‌های آزمون
- اجرای آزمونها و انتساب verdict که بصورت دستی انجام می‌شود.

آقای Andrews در مقاله‌اش برای پارتیشن بندی برنامه یا همان کلاستر کردن آن سه راه زیر را پیشنهاد کرد:

- User controlled: به این معنی که یک ویرایشگر در اختیار کاربر قرار می‌گیرد تا با استفاده از آن بتواند کلاسترها را مشخص کند. در این روش تعریف کلاسترها کاملا تحت کنترل کاربر و بصورت دستی می‌باشد.
- Full automation: این روش کاملا بصورت خودکار صورت می‌گیرد و از روشهای machine learning استفاده می‌کند.

[1] annotate

- Combination: در این روش بصورت خودکار یک پارتیشن بندی توسط ابزار انجام می‌شود و سپس کاربر آنرا تصحیح می‌کند. مثلا می‌توان از روش threshold استفاده کرد، بدین معنی که اگر تعداد لینکهای بین دو LWP[1] از یک مقدار threshold مشخص بیشتر باشد آندو در یک کلاستر خواهند بود. اما role based که هدف اصلی پارتیشن بندی می‌باشد، در این روش جایی ندارد.

در ابزار FSMWeb از روش سوم استفاده شده است، که یک روش نیمه خودکار است، و کاربر در آن نقش تصحیح کننده را بعهده دارد.

همانطور که قبلا گفته شد نمونه آزمون شامل مسیر آزمون و داده آزمون می‌باشد. برای ایجاد مسیر آزمون، از الگوریتمهای پیمایش گراف استفاده می‌شود. برای تولید مسیرهای در سطح FSMهای بالاتر مسیرهای FSMهای لایه زیری با هم ادغام می‌شوند. برای تولید داده آزمون از یک پایگاه داده مقادیر استفاده شده است که این پایگاه داده بصورت دستی و توسط کاربر ایجاد می‌شود. همانطور که ملاحظه می‌شود در این روش نیز میزان کاری که بصورت دستی انجام می‌شود تقریبا بالاست.

۴-۲-۲ ابزار WebUml و TestUml

آقای C.Bellettini در سال ۲۰۰۵ دو ابزار WebUml و TestUML را توسعه داد که WebUML یک ابزار مهندسی مجدد[2]برای تولید مدل از روی کد منبع می‌باشد. این ابزار کد منبع را بعنوان ورودی می‌گیرد و با استفاده از آنالیز کد منبع مدل ناوبری[3]سیستم را بصورت مدل UML تولید می‌کند.

[1] Logical Web Page

[2] reengineering

[3] navigate

TestUml از خروجی مرحله قبل استفاده کرده و اسکریپت آزمون را براساس پیمایش تصادفی[1] گراف تولید می‌کند، فایل log را برای تولید داده ورودی پیمایش کرده و از کاربر می‌خواهد تا داده‌های باقیمانده و نتایج مورد انتظار را بصورت دستی وارد کند. بعد از تولید نمونه‌های آزمون، آنها را اجرا و میزان پوشش را محاسبه می‌کند. اگر پوشش بدست آمده از پوشش مورد انتظار بیشتر باشد اجرا متوقف می‌شود در غیر اینصورت فرایند تولید و اجرای نمونه آزمون دوباره تکرار می‌شود. در واقع این روش ادغام کار آقای Ricca وElbaum می‌باشد که در مقایسه با روشهای قبل به مکانیزاسیون بیشتری نائل شده است. اما نکات ضعفی دارد که در قسمتهای آینده بیشتر بحث خواهد شد. [Bel04] [Bel05]

۵-۲-۲ آنالیز ایستای کد منبع

در سال ۲۰۰۸ آقای Wang روشی ارائه داد که مبنای کارش بر اساس آنالیز استاتیک کد منبع بود. بعبارتی کد منبع برنامه برای بدست آوردن اطلاعات مورد نیاز پردازش و آنالیز می‌شود. همانطور که می‌دانیم از تکنولوژیهای مختلفی برای پیاده‌سازی صفحات وب استفاده می‌شود اما Wang در مدل پیشنهادیش فقط بر روی صفحات JSP متمرکز شده است. ویژگیهای برنامه از کد منبع استخراج و بصورت مدل ناوبری نمایش داده می‌شود. در مدل ارائه شده نودها صفحات وب و لبه‌های جهت دار لینک بین صفحات می‌باشد. لبه‌ها با پارامترهای ورودی حاشیه نویسی می‌شوند. هر مسیر در این گراف نشاندهنده یک رشته اجرای صفحات توسط کاربر در حین ناوبری وب است.

پارامترهای ورودی که در مقاله Wang از آن با عنوان واسط[2] یاد شده است، با آنالیز کد منبع بدست می‌آید. پارامترهای ورودی بدون دانستن اطلاعات دامنه‌ای آن کمک چندانی به عمل آزمون نمی‌کند.

---

[1] Random walk

[2] interface

بهمین دلیل اطلاعات دامنه‌ای هر پارامتر ورودی با استفاده از نوعی الگوریتم کشف واسط از کد منبع استخراج می‌شود.

گراف ناوبری و واسط برای تولید نمونه‌های آزمون استفاده می‌شوند. نمونه آزمون شامل مسیر آزمون و داده‌های ورودی است. بنابراین قدم اول در ایجاد نمونه آزمون، تولید مسیر آزمون است. مسیر شامل یک رشته انتقال در گراف ناوبری و شرایط مسیر[1]می‌باشد. گام دوم تولید داده برای هر پارامتر ورودی مورد نیاز است. برای این منظور از بین داده‌های مرتبط با استفاده از اطلاعات دامنه‌ای استخراج شده، بصورت تصادفی داده آزمون انتخاب می‌شود.

چون از روش آنالیز ایستا استفاده شده است، ورودی فرم‌ها و اطلاعات دامنه‌ای بسیار دقیق تر از روشهای قبلی استخراج می‌شود. اما چون زبانهای برنامه‌نویسی و تکنولوژیهای زیادی برای پیاده‌سازی صفحات وب وجود دارد و این تکنولوژیها بسیار سریع در حال تغییر هستند، این روش عمومیت ندارد و فقط یک یا چندین تکنولوژی خاص را پشتیبانی می‌کند. همانطور که فقط صفحات JSP را پشتیبانی می‌کند. [Wan08]

۳-۲ مزایا و معایب روشهای ارائه شده:

۱-۳-۲ مشکلات ابزار ReWeb:

در این روش عملا فقط قسمتهای ساده کار بصورت خودکار انجام می‌شود و هیچ گونه خودکارسازی برای قسمتهای پیچیده کار انجام نشده است.

[1] Condition path

چون بصورت جعبه سفید عمل می‌کند. عمومیتش را از دست داده و وابسته به تکنولوژی خاصی شده است. بدلیل سرعت زیاد تغییر تکنولوژیها، عملا بعد از مدت کوتاهی ابزار ارائه شده غیر قابل استفاده می‌شود.

نویسنده ادعا کرده که تمام مسیرهای وب سایت آزمون می‌شود، اما به نظر ادعای اشتباهی است چون در مرحله تولید مدل صفحات دینامیک اصلا در نظر گرفته نمی‌شوند.

۲-۳-۲ مشکلات ابزار FsmWeb:

این روش تعدادی از مشکلات روش قبل را برطرف کرده است اما هنوز مشکل داده‌های ورودی کاملا برطرف نشده است. در این روش فرض شده ورودیها از روی پایگاه داده خوانده می‌شود، که چگونگی تولید پایگاه داده جای بحث دارد و نیاز به کار دستی زیادی است. نگاشت بین ورودیها و پایگاه داده نیز بصورت دستی انجام می‌شود.

۲-۳-۳ مشکلات روش مبتنی بر جلسه:

حالتی که رشته URL در هیچ کدام از جلسه‌ها وجود نداشته باشد، در نظر گرفته نشده است. برای تولید هر نمونه آزمون باید یکبار فایل log جستجو شود، که این عمل باعث کند شدن سرعت تولید نمونه‌های آزمون می‌شود. مسیرهای آزمون محدود به میرهای موجود در فایل log می‌شود و تمام مدل را پوشش نمی‌دهد.

۲-۳-۴ مشکلات روش آنالیز استاتیک کد منبع:

انواع مختلف زبانهای برنامه نویسی و تغییر سریع تکنولوژیها، باعث غیر قابل استفاده شـدن سـریع آن می‌شود. وابستگی به تکنولوژی و زبانهای برنامه‌نویسی خاص باعث کاهش فاکتور عمومیت ابزار می‌شود.

فصل ۳.روش آزمون مبتنی بر مدل پیشنهادی

۳–۱ مقدمه

همانطور که در فصل قبل ذکر شد آزمون مبتنی بر مدل بر اساس یک توصیف رسمی یا نیمه رسمی از SUT انجام می شود. در این روش، اسکریپت آزمون و یا داده های آزمون از روی مدل سیستم مورد آزمون تولید می شوند. منظور از مدل، یک توصیف رسمی یا نیمه رسمی از تمام یا قسمتی از سیستم می‌باشد. چنانچه چنین مدلی از سیستم در دسترس نباشد، باید به گونه‌ای آنرا استخراج کرد. در ابزار پیشنهادی فرض شده چنین مدلی موجود نیست و هدف بدست آوردن مدل ناوبری سیستم و انجام عمل آزمون بر مبنای مدل مذکور می‌باشد. مهمترین مشکل در بحث خودکارسازی آزمون برنامه‌های تحت وب پر کردن فرمهاست. قسمتهای پویای سایت غالبا در پشت فرمها مخفی هستند. ایده این پروژه برای پر کزدن خودکار فرمها استفاده از آنتولوژی است، که مفصلا در ادامه بخش توضیح داده می‌شود. به منظور مدل کردن نرم‌افزار ابتدا قسمتهای ایستای سایت با استفاده از انجام پرسش و پاسخ از سمت کلاینت مدل شده و سپس برای افزودن قسمتهای پویا به مدل از سه آنتولوژی کمک گرفته می‌شود. پس از اتمام فرایند مدلسازی در فاز بعد نمونه‌های آزمون ایجاد می‌شود. نمونه‌های آزمون شامل مسیر و داده آزمون می‌باشد.

مسیر آزمون بصورت خودکار بر مبنای سیاست پوشش مدل و با استفاده از الگوریتم‌های پیمایش گراف ایجاد می‌شود. سپس بر مبنای سیاست انتخاب مقادیر مرزی و با کمک سه آنتولوزی مذکور داده آزمون انتخاب می‌شود. بعد از ایجاد نمونه‌های آزمون وارد فازاجرا می‌شویم. در این پروژه از ابزار منبع باز Selenium[1] به منظور اجرای نمونه‌های آزمون استفاده شده است.

امکان دیگری که به پروژه اضافه شده است، ایجاد اسکریپت بر مبنای مسیر داده شده می‌باشد. بدین معنی که مسیر آزمون بهعنوان ورودی به ابزار داده شده و اسکریپتی بر اساس همان مسیر تولید می‌شود. برخی از نرم‌افزارهای تحت وب ساختار سناریو گرا دارند مانند سایتهای خرید الکترونیکی. با استفاده از اسکریپت مبتنی بر مدل امکان آزمون یک سناریو وجود ندارد. اما می‌توانیم از امکان دیگر ابزار بصورت زیر استفاده کنیم، مسیر سناریو را بعنوان ورودی به ابزار می‌دهیم، سپس اسکریپتی برای آزمون همان سنایو تولید می‌شود.

۲-۳ مهندسی مجدد نرم افزارهای کاربردی تحت وب

هدف اصلی در فرایند مهندسی مجدد ایجاد نمایش ساده‌تری از سیستم می‌باشد، لازم است چنین نمایشی مدل ساختاری و رفتاری سیستم را نمایش دهد. انتظار می‌رود مدل ایجاد شده نگاشت بین اجزاء ساختاری و رفتاری سیستم را فراهم کرده و تمام اطلاعات مورد نیاز را در سطوح مختلف نمایش دهد. اولین و بزرگترین جزء در فرایند مهندسی مجدد برنامه‌های تحت وب، صفحات هستند. صفحات وب به دو دسته صفحات سرور و صفحات کلاینت تقسیم می‌شوند. به صفحاتی که بر روی سرور استقرار دارند صفحات سرور و به آنهایی که در پاسخ به درخواست کلاینت فرستاده می‌شوند صفحات کلاینت گویند. صفحات کلاینت به دو دسته ایستا و پویا تقسیم می‌شوند. صفحاتی که محتوایشان ثابت است و به شیوه

[1] Selenium is a suite of tools to automate web app testing across many platforms. http://seleniumhq.org

پایا[1]ذخیره می‌شوند صفحات ایستا و آنهایی که بصورت موردی ایجاد شده و محتوایشان دائما در حال تغییر است صفحات پویا می‌گویند.

ساختار برنامه‌های کاربردی تحت وب به دو صورت زیر قابل استخراج است:

تکنیک جعبه سفید یا آنالیزکد منبع[2]: در این روش کد منبع برای استخراج مدل پویش و آنالیز می‌شود. معمولا ابزارهای توسعه یافته با این روش برای برخی زبانها و تکنولوزیهای خاص توسعه می‌یابند. بهمین دلیل غالبا عمومیت چنین ابزارهایی پایین است و وابسته به تکنولوژیهای خاص می‌باشد. بدلیل رشد و تغییر سریع تکنولوزیهای وب غالبا چنین ابزارهایی نیاز به نگهداری دائمی دارند، در غیر اینصورت خیلی سریع کارایی خود را از دست می‌دهند.

تکنیک جعبه سیاه یا آنالیز از سمت کلاینت[3]:این تکنیک در سمت کلاینت انجام می‌شود و ساختار سایت را با ایجاد درخواست‌ها و بررسی پاسخ‌ها بدست می‌آورد. لازم به ذکر است که صفحات پویا و ایستا بدین روش قابل تشخیص نیستند اما بصورت غیر رسمی می‌توان گفت وقتی یک فرم پر شده و به شمت سرور فرستاده می‌شود، صفحه مقصد پویا است و محتوایش وابسته به داده‌هایی است که کاربر در فرم وارد کرده است.

در ابزار ارائه شده از تکنیک دوم یا همان تکنیک جعبه سیاه استفاده شده است.

همانطور که می‌دانیم یک صفحه ممکن است به چندین فریم[4]تقسیم شود. در این حالت ارتباطات بین صفحات به دو دسته تقسیم می‌شود، لینکهای بین دو صفحه و ارتباطات بین اجزاء یک صفحه. ارتباطات بین صفحات شامل: ارتباط بین یک فرم و صفحه سرور، ارتباط redirect بین اسکریپت و صفحه

[1] Persistent way

[2] Source code analysis

[3] Client side analysis

[4] Frame

دیگر، ارتباط include بین یک اسکریپت کلاینت و یکی از ماژولهای کلاینت و ارتباط بین اجزاء صفحات می‌باشد. همانطور که قبلا اشاره شد، یک صفحه می‌تواند شامل هر تعدادی فرم باشد. هر فرم با ورودیهایش توصیف می‌شود، ورودیها توسط کاربر وارد می‌شوند و محتوای صفحه مقصد بسته به ورودیها تغییر می‌کند. اگر بخواهیم تمامی مسیرها را آزمون کنیم باید تمامی ورودیها را تولید و امتحان کنیم.

۳-۳ ایجاد مدل

مدل آنالیز شده ای که در اینجا مورد بحث قرار می‌گیرد، بیشتر بر روی الگوی تعاملات و ناوبری سیستم متمرکز شده است. در زیر یک برنامه کاربردی وب با اطلاعاتی که از یک سرور قابل دسترسی اند در نظر گرفته شده است. اسنادی که از سرورهای دیگر دستیابی می‌شوند، صفحات خارجی نامیده می‌شوند و ما در آنالیزمان از آنها صرف نظر می‌کنیم.

یک صفحه وب شامل اطلاعاتی است برای نمایش به کاربر و لینکهایی به صفحات دیگر. ورودیهای کاربران با استخراج فرمها بدست می‌آید. همانطور که گفتیم یک صفحه می‌تواند شامل هر تعدادی فرم باشد و فرمها با متغیرهای ورودیشان مشخص می‌شوند، که مقادیر این متغیرها توسط کاربر هنگام اجرا فراهم می‌شود. مقادیری که توسط فرمها جمع آوری شده اند توسط لینک submit به سرور فرستاده می‌شود، که غالبا مقصد این نوع لینک‌ها یک صفحه پویاست. فرمی را در نظر بگیرید که علاقه‌مندی کاربر را پرس و جو کرده و اطلاعاتی درباره علاقه‌مندی او نمایش می‌دهد. بعنوان مثال اگر علاقه‌مندی کاربر ورزش باشد، اطلاعاتی درباره ورزش نمایش داده می‌شود. نتیجه اینکه برای استخراج تمامی مسیرها باید تمام اطلاعات مورد نیازی که متغیرها می‌توانند بگیرند را فراهم کنیم. فرایند مدلسازی برنامه کاربردی وب طی دو فاز زیر انجام می‌شود:

آنالیز ایستا: در این فاز ساختار ایستای برنامه تحت وب شامل صفحات ایستا، لینکهای ایستا و . . . مدل می‌شوند

آنالیز پویا: همانطور که توضیح داده شد، چون صفحات پویا در پشت فرمها مخفی هستند، در این فاز مقادیر ورودی فرمها تولید و حاصل به صفحات پویا فرستاده می‌شود. بصورت غیر رسمی این فاز را پر کردن فرمها می‌نامیم.

۱-۳-۳ آنالیز ایستا[1]

در این فاز ابزار ارائه شده از آدرس داده شده شروع کرده و تمامی مسیرهای اجرایی موجود در فضای حالت قابل دسترس از صفحه مذکور را مرور می‌کند. هر صفحه دیده شده توسط جزء دیگری از ابزار برای خطاهای احتمالی کنترل می‌شود، در صورت مشاهده خطا، log می‌شود. اگر در حین پویش چرخه مشاهده شد، صفحه جاری صرف نظر می‌شود، در غیر اینصورت اعمال ممکن از صفحه جاری برای بررسی‌های آتی ضبط و نگهداری می‌شود. سپس اعمال ضبط شده به ترتیب و مطابق با الگوریتم زیر پردازش می‌شوند:

- با دادن یک لینک، HREF لینک در آدرس مرورگر قرار می‌گیرد. اگر لینک مذکور رویداد onclick داشته باشد، هندلر[2]مربوطه فراخوانی می‌شود و در صورتی که مقدار درست را برگرداند، مرورگر به آدرس HREF تغییر موقعیت می‌دهد.

[1] Static analusis

[2] handler

- با دادن فرم، با استفاده از استراتژی بخش بعد برای ورودی‌های فرم مقادیری انتخاب شده و در محل مورد نظر قرار می‌گیرند و هندلر onsubmit (در صورت وجود) فراخوانی شده و سپس فرم فرستاده می‌شود.
- با دادن یک المان فعال فرم، هندلر رویداد مربوطه فراخوانی می‌شود.

۲-۳-۳ پر کردن خودکار فرم

اصلی ترین موضوع در خودکارسازی فرایند آزمون مبتنی بر مدل برای برنامه‌های تحت وب پر کردن فرمها است. فرمی را در نظر بگیرید که نام کاربر را گرفته و یک صفحه خوشامدگویی شخصی نمایش می‌دهد. ابزار خودکار قادر نیست بفهمد چه ورودیهایی کافیست تا حالات بعدی این صفحه را پوشش دهد، حتی اگر تعداد این حالات کم باشد.

بعوآن یک مثال استاندارد فرمی را در نظر بگیرید که کلمه عبور و رمز کاربر را دریافت می‌کند، بدون داشتن اطلاعات اضافی هیچ ابزاری قادر به تشخیص حالات بعدی آن صفحه نمی‌باشد بجز حالتی که پیغام می‌دهد کلمه عبور و رمز اشتباه وارد شده است. بنابراین نیاز به اطلاعات اضافی است تا ابزار بتواند در میان فرمها نفوذ و فضای حالات را پوشش دهد. در اینجا دو سوال مطرح می‌شود، در چه سطحی نیاز به اطلاعات اضافی داریم و چگونه از این اطلاعات باید استفاده کرد.

برخی از داده‌هایی که کاربران در فرمها وارد می‌کنند از اهمیت بیشتری برخوردار است و باید ذخیره و در محلی امن مانند پایگاه داده نگهداری شود، اما برخی دیگر از اطلاعات اهمیت کمتری دارند و بصورت موقت برای پردازشهای آتی ذخیره می‌شوند. داده‌های مهمی که در محل پایا ذخیره می‌شوند همان اطلاعات اضافی‌ای هستند که درباره آنها صحبت شد. اما سوالی که در اینجا مطرح می‌شود اینست که چگونه از این داده‌ها برای پر کردن فرمها استفاده می‌شود و چگونه یک متغیر ورودی موجود در فرم به

یک فیلد در پایگاه داده نگاشت میشود. اگر اطلاعات آن متغیر در پایگاه داده سیستم ذخیره نشود چگونه برای آن متغیر داده تولید می‌شود. در بخش بعدی به این سوالات پاسخ داده می‌شود.

۳-۳-۳ استفاده از آنتولوژی[1]برای تولید داده

آنتولوژی روش مفیدی برای نگهداری و به اشتراک گذاری دانش دامنه بمنظور تسهیل فرایند ایجاد اسکریپت آزمون است. آنتولوژی می تواند بعنوان ابزار مفیدی جهت خودکارسازی آزمون بکار رود. با توجه به اینکه آنتولوژی، قابلیت ذخیره دانش مربوط به یک حوزه در قالب قابل استفاده توسط ماشین را دارد، می توان با استفاده از آن برخی از قسمتهای آزمون نرم افزار را خودکار نمود. در سیستم مورد نظر از ایده آزمون مبتنی بر آنتولوژی استفاده شده است.

در این پروژه از سه آنتولوژی به شرح زیر استفاده شده است:

آنتولوژی عمومی: بیشتر وب سایتها اطلاعاتی مشابه را توسط فرمها از کاربران می‌گیرند، مانند اطلاعات شخصی، اطلاعات مربوط به حسابهای بانکی، اطلاعات آموزشی و غیره. می‌توان گفت تقریبا ۸۰ درصد اطلاعاتی که کاربران در وب سایتهای مختلف وارد می‌کنند اطلاعاتی عمومی و مشابه مانند سن و جنس و نام و . . . هستند، که ما اقدام به جمع آوری این اطلاعات از سایتهای مختلف کردیم. بدین منظور حدود ۱۰۰ فرم از سایتهای مختلف بارگیری و نام متغیرهای متناظر با المانهای متنی فرمها در فایلی مخصوص ذخیره شد. سپس به منظور حذف کلمات مترادف فایل مذکور پردازش و در نهایت فایلی شامل تعدادی کلمه که هیچ کدام از آنها مترادف نبودند، تولید شد. آنتولوژی عمومی از روی فایل مذکور ایجاد و منابع[2]و مفاهیم[1]آن با محدودیتها[2]و قیودی دقیق تعریف شدند. بعنوان مثال کلمه 'age'، المان متنی در

[1] Ontology

[2] Resource

فرم ثبت نام سیستم دانشجویی را در نظر بگیرید. دو محدودیت که برای مفهوم 'age' در آنتولوژی عمومی تعریف می‌شود min و max می‌باشد که به ترتیب برابر ۱۰ و ۸۵ تعریف می‌شود.

لغتنامه مانند wordnet: از این آنتولوژی برای درک کلمات مترادف و متضاد استفاده می‌شود. WordNet یک پایگاه داده بزرگ لغوی از کلمات انگلیسی است. کلمات، فعلها، صفات و قیود به گروههای مترادف که هر کدام بعنوان یک مفهوم استفاده می‌شود، تقسیم شده است. لغت نامه، می تواند در تشخیص کلمات مترادف و متضاد بکار رود. بعنوان مثال، اگر در یک فرم، فیلدی با نام gender وجود داشته باشد، اما در آنتولوژی عمومی، مفهومی با این عنوان موجود نباشد. برنامه مورد نظر می‌تواند با استفاده از لغتنامه، استنتاج کند که کلمه gender مترادف کلمه sex است. در نتیجه چنانچه در آنتولوژی، مفهوم sex تعریف شده باشد، می تواند به آن نگاشت شود.

آنتولوژی سیستم: این آنتولوژی از روی پایگاه داده سیستم ایجاد می‌شود، داده‌های ذخیره شده در پایگاه داده به نمونه ها در آنتولوژی سیستم تبدیل می‌شود. لازه به ذکر است که برای تولید آنتولوژی از روی پایگاه داده سیستم از قواعد ارائه شده در [Etm09] استفاده کردیم.

حال می‌خواهیم ببینیم چگونه از سه آنتولوزی فوق برای پر کردن فرم‌ها استفاده می‌شود. یک المان متنی در یک فرم را در نظر بگیرید. برای پر کردن المان، ابتدا ابزار سعی می‌کند آنرا به یکی از مفاهیم موجود در آنتولوژی سیستم نگاشت کند، در آنصورت داده‌ای از میان نمونه‌های موجود در آنتولوژی سیستم بعنوان نمونه آزمون انتخاب می‌شود. اگر چنین نگاشتی امکان پذیر نباشد، یعنی مفهومی متناظر با چنین المانی در آنتولوژی سیستم موجود نباشد، آنگاه المان به یکی از مفاهیم موجود در آنتولوژی

[1] Concept

[2] Constraint

عمومی نگاشت می‌شود. سپس از محدودیتهای تعریف شده برای آن مفهوم استفاده شده و بـا اسـتفاده از سیاست مقدار مرزی[1] داده‌ای برای المان مورد نظر تولید می‌شود.

برای توضیح بیشتر مفهوم 'age' را در آنتولوژی عمومی در نظر بگیرید اگـر دو محـدودیت مینـیمم و ماکزیمم برای آن تعریف شده باشد با استفاده از سیاست مقادیر مرزی داده‌هـای min+1,min,mid,max-1,max انتخاب می‌شوند. برای توضیح بیشتر مفهوم 'gender' را در نظر بگیرید، دو مقداری که این متغیر می-تواند بگیرد در آنتولوژی عمومی تعریف شده و شامل 'female' , 'male' می‌باشد، که یکی از آندو به تصادف انتخاب می‌شود.

نحوه برخورد ابزار ارائه شده با هر کدام از انواع المانها متفاوت است، در زیر هر کدام از انـواع المانهـا و نحوه برخورد با آنها توضیح داده می‌شود.

Text Box: ابتدا متغیر متناظر با المان به یکی از مفاهیم آنتولوژی عمومی نگاشت می‌شـود. دو حالـت زیر بسته به شرایط بوجود می‌آید:

- مفهوم متناظر با المان در آنتولوژی عمومی بـه یکـی از مفـاهیم آنتولـوژی سیسـتم نگاشـت می‌شود، در اینصورت نمونه‌های آزمون از آنتولوزی سیستم انتخاب می‌شود.
- امکان نگاشت مفهوم متناظر در آنتولوژی عمومی به یکی از مفاهیم آنتولوژی سیسـتم وجـود ندارد. در اینصورت از محدودیتهای موجود در آنتولوژی عمومی استفاده شده و نمونه آزمـون تولید می‌شود.

همانطور که گفته شد نمونه آزمون با استفاده از معیار پوشش مرزی انتخاب مـی‌شـود. نحـوه انتخـاب داده نسبت به نوع متغیر متفاوت است، بعنـوان مثـال اگـر متغیـر از نـوع عـددی باشـد، آنگـاه داده‌هـای min,min+1,mid,max-1,max بعنوان نمونه آزمون انتخاب می‌شوند. اما اگر نوع متغیر از نوع متنی باشد

[1] Boundary value policy

آنگاه یک متن با طول کمتر از حداکثر طول، یک متن با طولی برابر با طول حداکثر و یک متن با طول بیشتر از طول حداکثر انتخاب می‌شود.

Text Area: این المان به منظور گرفتن یک متن یا چندین پاراگراف از کاربر استفاده می‌شود. نمونه‌های آزمونی که برای این المان تولید می‌شوند، عبارت است از: یک متن با طول برابر، یک متن با طول بیشتر و متنی با طولی کمتر از طول حدکثر.

Radio group: برای این منظور یکی از دکمه‌های رادیویی[1] به صورت تصادفی انتخاب می‌شود.

Checked Box: دو مقداری که این المان می‌تواند بگیرد on و off هستند، که به تصادف یکی از آنها انتخاب می‌شود.

Select: مقادیری که به این المان می‌توان انتساب داد، در تگ option مربوط به آن مشخص شده است، که یکی از آنها به صورت تصادف انتخاب می‌شود.

همانطور که قبلا اشاره کردیم اگر یک متغیر در آنتولوژی سیستم موجود باشد، نمونه‌های آزمون از داده‌های ذخیره شده در آنتولوژی سیستم انتخاب می‌شود در غیر اینصورت از محدودیتهای تعریف شده در آنتولوژی عمومی برای تولید نمونه آزمون استفاده می‌شود.

۴-۳ آزمون

به دو روش می‌توان برنامه‌های تحت وب را آزمود، روش ایستا و روش پویا. در روشهای ایستا نیاز به اجرا نیست و صفحه html برای یافتن خطهای احتمالی و برخی آنومالی‌ها پویش می‌شود. در روش آزمون پویا برداری از داده‌های ورودی (نمونه آزمون) بر روی سیستم آزمون شده و نتیجه بدست آمده با نتیجه

[1] Radio Button

مورد انتظار مقایسه می‌شود. سپس بسته به نتیجه مقایسه مقدار fail یا success به نمونه آزمون منسوب می‌شود.

نمونه آزمون برنامه‌های تحت از یک رشته URL و داده‌های ورودی تشکیل شده است که داده‌های ورودی برای صفحاتی است که شامل فرم می‌باشند. برای اعمال معیار پوشش مدل[1]مسیر آزمون از روی مدل تولید می‌شود و سپس برای هر مسیر در صورت نیاز داده‌های ورودی انتخاب می‌شود. بنابراین می‌توان گفت انتخاب مسیر آزمون از انتخاب داده آزمون جداست و می‌تواند مستقلا و بصورت خودکار انجام شود. مشکلترین قسمت خودکارسازی، انتساب داده به متغیرهای ورودی است که قبلا در دو بخش پر کردن فرمها و آنتولوژی به تفصیل شرح داده شد.

در فرایند اجرا URL ها به ترتیب از سرور درخواست می‌شوند و کد جواب[2]برای گزارش گیری‌های آتی ذخیره می‌شود. مدل ناوبری استخراج شده از مرحله قبل و ۳ آنتولوژی مذکور ورودی این جزء MBTester می‌باشد. در این فاز از اجرا مدل برای تولید رشته URLها[3]بکار گرفته می‌شود. به این ترتیب که با کمک برخی الگوریتم‌های پیمایش گراف و بر مبنای استراتژی پوشش مدل مسیرهای آزمون انتخاب می‌شوند. سپس با کمک ۳ آنتولوژی ذکر شده و با استفاده از تکنیک پوشش مقادیر مرزی و تقسیم بندی داده‌های هم ارز[4]داده‌های ورودی تولید می‌شوند. اسکریپت آزمون به فرمتی ایجاد می‌شود که توسط ابزار منبع باز Selenium قابل اجرا باشد. در این پروژه از Selenium به منظور اجرای نمونه‌های آزمون استفاده می‌شود. لازم به ذکر است که اسکریپت ایجاد شده علاوه بر نمونه آزمون شامل oracle نیز می‌باشد و selenium می‌تواند امکانات گزارشگیری از تعداد گامهایی که با موفقیت اجرا شده‌اند و تعداد

[1] Model coverage criterion

[2] Response code

[3] URL sequence

[4] Equivalent partitioning

گامهایی که با شکست مواجه شده است را در اختیار ما قرار می‌دهد. برای اجرای اسکریپت آزمون، فایل شامل اسکریپت در Selenium بارگذاری شده و سپس اجرا می‌شود.

۴-۴ ایجاد نمونه آزمون بر اساس مسیر داده شده

MBTester امکانی دارد که مسیر آزمون را به فرمتی خاص از ورودی می‌گیرد و نمونه آزمون را بر اساس آن مسیر ایجاد می‌کند. می‌توان از این اسکریپت برای منظورهای زیر استفاده کرد:

- آزمودن یک سناریو: بدین ترتیب که مسیر سناریو را بعنوان ورودی به ابزار می‌دهیم . سپس اسکریپتی برای آزمودن همان سناریوی خاص ایجاد می‌شود.
- می‌توان مسیر آزمون را از روی فایل log سرور انتخاب کرد. در اینصورت اسکریپت تولید شده کاملا مطابق با عملکرد کاربر می‌باشد. چنین اسکریپتی کاملا مناسب آزمون بار[1] می‌باشد و می‌توان برای این منظور از آن استفاده کرد.

دلیل استفاده از این امکان بیشتر برای نرم‌افزارهای سناریو گرا می‌باشد. اسکریپتی که توسط روش مبتنی بر مدل ایجاد می‌شود، تمامی لینکها را تا سطح خاصی که کاربر بعنوان ورودی می‌دهد، آزمون می‌کند و اینگونه نیست که مسیر یک سناریو را دنبال کند. بهمین دلیل نمی‌توان صحت یک سناریو را با این روش تایید کرد و باید از روش مبتنی بر مسیر داده شده استفاده شود.

[1] Load test

فصل ۴. نتیجه‌گیری و پیشنهادات

در این فصل در ابتدا ابزار پیاده‌سازی شده ارزیابی می‌شود و در ادامه پیشنهادات برای کـار بیشـتر در این زمینه ارائه خواهد شد. این ابزار در ادامه ابزارهای موجود می‌باشد. در ابتدا تفاوت بین این ابزار با دیگر ابزارها بررسی می‌شود و سپس تاثیر این تغییر مورد بحث قرار می‌گیرد.

۴–۱ تفاوتها با کارهای قبلی

بزرگترین تفاوت بین این ابزار با کارهای قبلی افزایش میزان خوکارسازی است. در این ابزار از چندین آنتولوژی برای پیاده‌سازی خودکار عمل آزمون استفاده شده که ایده ای نو می‌باشد. مهمترین مشکل برای آزمون برنامه‌های تحت، صفحات دینامیک می‌باشد. صفحات دینامیک غالبا در پاسخ به submit یـک فـرم ایجاد می‌شوند. چنین صفحاتی وابسته به داده هایی هستند که کاربر هنگام اجرا در فرم وارد می‌کند. این صفحات در پشت فرمها مخفی هستند و برای آنالیز و پویش آنها نیاز به تولید داده برای فرمها مـی‌باشـیم. برای اینکه عمل آزمون تمام مسیرها را پوشش دهد باید داده‌های زیادی بصورت هوشمندانه تولیـد شـود. ابزارهای قبلی هیچ ایده‌ای برای تولید خودکـار داده‌هـای آزمـون و در نتیجـه پـر کـردن خودکـار فرمهـا نداشتند. یعنی عملا تحلیل و مدلسازی قسمتهای پویای کار حجم کار دستی زیادی را به کـاربر متحمـل

می‌کرد. اما در روش ارائه شده و با استفاده از مفاهیم آنتولوژی بهبود قابل توجهی در زمینه خودکارسازی انجام گرفته است.

۴-۲ مزایای ابزار پیاده سازی شده

محققان تاکنون نشان داده‌اند که آزمون مبتنی بر مدل برای برنامه‌های کوچک و تعبیه شده مناسب است. برای برنامه‌های بزرگ مانند برنامه تحت وب ابزارهای مختلفی پیاده‌سازی شده که در هر کدام از روش مختلف و بعضا مشابهی استفاده شده است. چیزی که در تمام این ابزارها مشابه بود درجه پایین خودکارسازی در تحلیل و مدلسازی قسمتهای دینامیک سیستم بود. این پروژه نشان داد که آزمون مبتنی بر مدل با میزان خودکارسازی بالا برای برنامه‌های تحت وب نیز مناسب است.

- در این پروژه امکان ایجاد اسکریپت از روی فایل log جلسه وجود دارد. تفاوت این اسکریپت با اسکریپت بدست آمده از پیمایش مدل در این است که در این اسکریپت فقط صفحاتی آزمون می‌شوند که در فایل log وجود دارد. فایل log عملکرد کاربر را ذخیره می‌کند. بهمین دلیل اسکریپت بدست آمده از فایل جلسه بسیار شبیه به کار کاربر می‌باشد. این اسکریپت برای زمانی که می‌خواهیم عملکرد سرور را در برابر آزمون لود بسنجیم بسیار مناسب است.
- می‌توانیم از یک اسکیپت آزمون ولی با چندین داده آزمون استفاده کنیم. داده‌های آزمون بر اساس سیاست معیار مقاددیر مرزی[1] و با استفاده ار آنتولوژی بصورت خودکار تولید می‌شوند. مثلا فرمی را در نظر بگیرید که کلمه عبور و رمز کاربر را وارد می‌کند و فرم را submit می‌کند، یک نمونه آزمون با کلمه عبور و رمز درست ایجاد می‌شود. نمونه دیگری با کلمه و رمز نادرست ایجاد می‌شود.

[1] Boundary value coverage criterion

۴-۳ ارزیابی و نتایج

همانطور که دیدیم فرایند آزمون دستی بعد از انتخاب نقشه آزمون از ۴ مرحله تشکیل شده است. ۱- انتخاب مسیر ۲- انتخاب داده ۳- اجرای نمونه‌های آزمون ۴- انتساب verdict. جدول ۴-۱ فرایند آزمون دستی و دیگر ابزارهای آزمون مبتنی بر مدل و روش پیشنهادی این پروژه را از نظر میزان خودکار بودن در ۴ مرحله فوق با هم مقایسه می‌کند.

ابزارهای موجود	طراحی آزمون		اجرای آزمون	انتساب verdict	تعداد فرایندهایی که خودکار انجام می‌شوند (از ۴
	انتخاب مسیر	انتخاب داده			
آزمون دستی	دستی	دستی	دستی	۰	۰
ابزار FSMWeb	خودکار	دستی	خودکار	۲	۲
ابزار ReWeb	خودکار	دستی	خودکار	۲	۲
ابزار TestUML	خودکار	دستی	خودکار	۳	۳
ابزارهای	دستی	دستی	خودکار	۲	۲
مدل پیشنهادی	خودکار	نیمه	خودکار	بیش از ۳	بیش از ۳ فرایند

جدول ۴-۱ مقایسه ابزار پیشنهادی با دیگر ابزارها از نظر درجه خودکارسازی

همانطور که مشاهده می‌شود می‌توان نتیجه گرفت که هر کدام از ابزارهای موجود چه میزان از کار را بصورت خودکار انجام می‌دهند، که در ستون آخر جدول ۴-۱ میزان خودکار بودن هر فرایند آمده است. همانطور که در جدول ۴-۱ پیداست مدل پیشنهادی بیشترین میزان خودکارسازی را دارد.

حال می‌خواهیم ببینیم که چند درصد فرایند انتخاب داده آزمون خودکار شده است. هر چه میزان خودکار شدن در فرایند انتخاب داده آزمون بیشتر باشد میزان خودکارسازی ابزار پیشنهادی به ۱۰۰٪

نزدیک می‌شود. در فصل بعد میزان خودکارسازی در فاز انتخاب داده آزمون برای دو نرم‌افـزار zen cart و webCalendar بررسی می‌شود.

۱-۳-۴ فرایند انتخاب داده آزمون:

همانطور که در فصل ۳ شرح داده شد، یک نمونه آزمون شامل رشته ای از URLها بعلاوه داده آزمون می‌باشد. رشته URL با پیمایش گراف با استفاده از نقشه آزمون انتخاب شده، تولید می‌شود. برای انتخاب یا ایجاد داده آزمون از سه آنتولوژی استفاده می‌شود. آنتولـوژی اول آنتولـوژی عمـومی اسـت کـه شـامل کلمات عمومی است. آنتولوژی دوم آنتولوژی است که از روی پایگاه داده ایجاد می‌شود، داده‌هـای ذخیـره شده در پایگاه داده بصورت نمونه در آنتولوژی ذخیره می‌شوند. آنتولوژی سوم لغتنامه WordNet می‌باشد، برای درک کلمات مترادف و متضاد. همانطور که قبلا گفتیم برخی از مقـادیری کـه کـاربران در فرمهـای مختلف یک سایت وارد می‌کنند در یک محل مانند پایگاه داده ذخیره می‌شود و برخی از داده‌ها در هـیچ جایی ذخیره نمی‌شوند. در ابتدا بین مفاهیم موجود در آنتولـوژی عمـومی و آنتولـوژی سیسـتم نگاشـتی صورت می‌گیرد. هنگام پر کردن یک فرم ابتدا فیلدهای موجود در فرم به مفاهیم آنتولوژی عمومی نگاشت می‌شود.

برای مثال، اگر یک فیلد بر روی یک فرم، با عنوان zip code موجود باشد، چنانچه مفهـوم zip code در آنتولوژی عمومی موجود باشد و قیود و محدودیت های آن بخوبی بیـان شـده باشـد، دو حالـت اتفـاق می‌افتد:

حالت اول وقتی است که این مفهوم به هیچ کدام از مفاهیم آنتولوژی سیسـتم نگاشـت نشـده باشـد. بدین معنی که مقدار این فیلد در پایگاه داده ذخیره نشده است. آنگاه برنامه تولید کننده داده‌های آزمون،

می تواند با استفاده از قیود و محدودیتهای آنتولوژی عمومی، داده‌های مناسبی با استفاده از معیار پوشش مقادیر مرزی تولید نماید و عملکرد سیستم را در برابر آنها بسنجد.

حالت دوم وقتی است که این مفهوم به یکی از مفاهیم آنتولوژی سیستم نگاشت شده باشد، آنگاه یکی از نمونه‌های موجود در آنتولوژی سیستم بعنوان داده آزمون و با استفاده از الگوریتم تصادفی انتخاب می‌شود.

ممکن است در آنتولوژی عمومی مفهوم zip code را داشته باشیم، اما بر روی فرم، فیلدی تحت عنوان postal code داشته باشیم، از لغت نامه برای نگاشت postal code به مفهوم zip code آنتولوژی عمومی استفاده می‌شود.

۴-۳-۲ Case study

در این قسمت یک بررسی موردی بر روی دو نرم‌افزار منبع باز به نام‌های webCalendar و Zen Cart انجام می‌شود، که به ترتیب در ادامه فصل بحث می‌شوند.

:WebCalendar

برنامه تحت وب منبع باز[1] webCalendar برای این منظور در نظر گرفته شده است این نرم‌افزار از آدرس http://webcalendar.sourceforge.net قابل بارگیری می‌باشد. webCalendar برنامه‌ای به زبان php است که یک تقویم را برای یک کاربر یا گروهی از کاربران اینترانت فراهم می‌کند. خاصیت پویایی زیاد، پایگاه داده و فرمهای متنوع و قوی آن دلیل انتخاب آن می‌باشد.

برنامه WebCalendar از یک پایگاه داده mysql به نام intranet استفاده می‌کند، که این پایگاه داده از ۲۹ جدول برای ذخیره اطلاعات استفاده می‌کند. اطلاعاتی که در پایگاه داده ذخیره می‌شود شامل

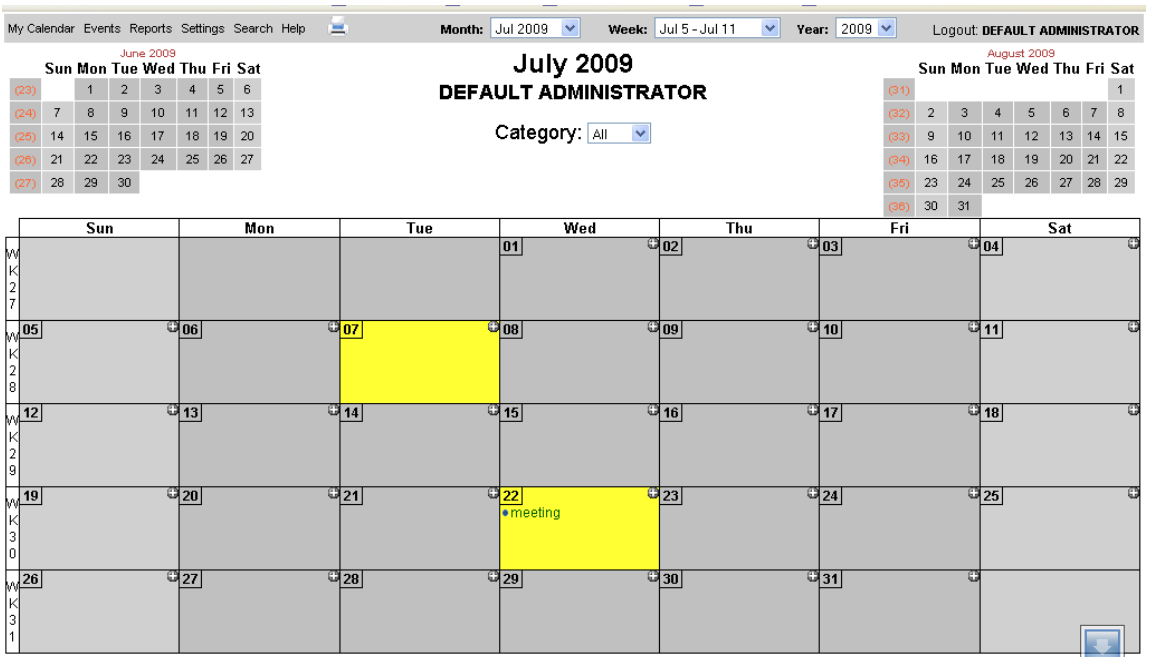
[1] Open source

اطلاعات مربوط به تنظیمات سیستم، گروههای کاربری، کاربران، وقایع، دیدها، یادآوریها و ... می‌باشد. هنگام ورود به سیستم کاربر باید کلمه عبور و رمز خود را وارد نماید. کلمات عبور و رمز کاربران در جدول webcal_user پایگاه داده ذخیره می‌شود، لازم به ذکر است که کلمه رمز بصورت انکد شده ذخیره

شکل ۴-۱ فرم login به سیستم

می‌شود.

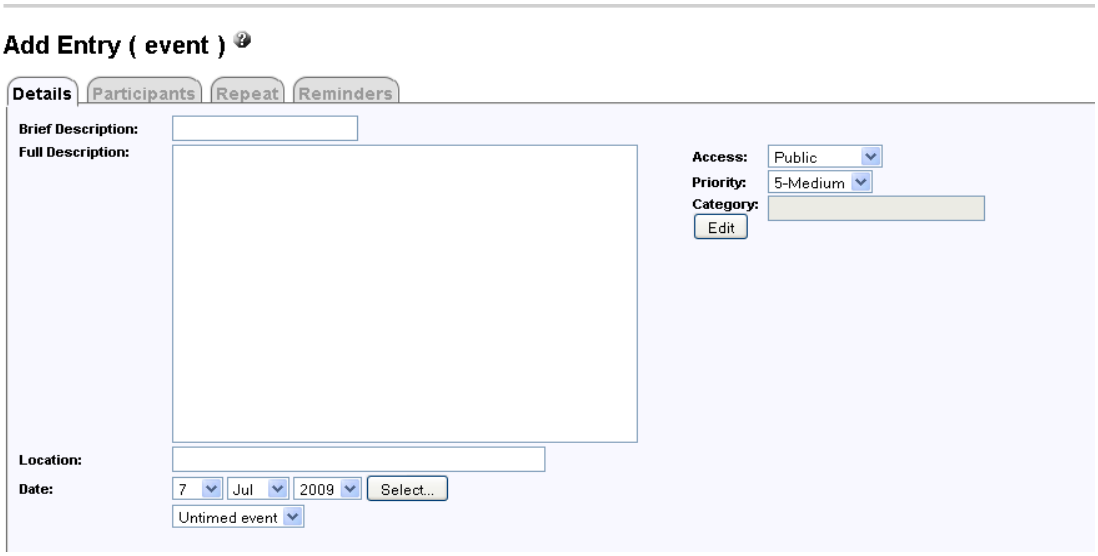
شکل ۴-۲ نمایی کلی از برنامه WebCalendar را نمایش می‌دهد. تقویم ۷ ژوئیه را نشان می‌دهد و یک رویداد با عنوان meeting در ۲۲ ژوئیه قابل مشاهده می‌باشد.



شکل ۴-۲ نمایی کلی از برنامه WebCalendar، یک رویداد به نام meeting در ۲۲ ژوئیه دیده می‌شود.

یکی از مهمترین فرمهایی که در نرم‌افـزار webCalendar پـر مـی‌شـود، فـرم افـزودن رویـداد جدیـد می‌باشد، که در شکل ۴-۳ نشان داده شده است.

همانطور که در فصل ۳ شرح داده شد، مشکل اصلی در مواجهه با المانهای text input است. همانطور که می‌بینیم فرم add entry شامل دو text box بـه نامهـای location و brief description و یـک text area به نام full description می‌باشد. سایز المـان location ۵۵، سـایز brief descrioption برابـر ۲۵ و سایز full description برابر ۵۰×۱۵ به معنی ۱۵ سطر ۵۰ ستون می‌باشد. المان‌های مورد نظـر بایـد بـه یکی از مفاهیم آنتولوژی عمومی نگاشت شود، میزان خودکارسازی ابزار رابطه مستقیمی با عمل نگاشت در این مرحله دارد. هر چه در عمل نگاشت کاربر نقش کمتری داشـته باشـد، میـزان خودکارسـازی افـزایش می‌یابد. همانطور که قبلا شرح داده شد. ابتدا ابزار بصورت خودکار المان مورد نظر را بـه یکـی از مفـاهیم آنتولوژی نگاشت می‌کند و کاربر انتخاب ابزار را در صورت لزوم تصحیح و سپس تایید می‌کند. متریکی که برای خودکارسازی استفاده شده بصورت زیر تعریف می‌شود:



شکل ۳-۴ فرم افزودن رویداد جدید (add entry)

$$\text{میزان خودکارسازی} = \frac{\text{تعداد المانهایی که درست نگاشت شده}}{\text{تعداد کل المانها}}$$

جدول ۴-۲ میزان خودکارسازی را برای هر فرم در برنامه webCalendar نشان می‌دهد.

نام فرم	تعداد کل المانها	تعدا المانهای اشتباه نگاشت شده	میزان خودکارسازی
login	۲	۰	٭۵۰٪
Add Entry	۱۳	۱	۹۲٪
Delete Events	۷	۰	۱۰۰٪
Import	۵	۱	۸۰٪
Export	۱۳	۰	۱۰۰٪
search	۱	۰	۱۰۰٪
میانگین			۸۵٪

جدول ۴-۲ میزان خودکارسازی انتخاب داده به تفکیک هر فرم

٭همانطور که قبلا ذکر شد password در پایگاه داده بصورت انکد ذخیره می‌شود، پس حتی اگر عمل نگاشت با موفقیت انجام شود، داده‌ای که از آنتولوژی سیستم برای متغیر password انتخاب می‌شود، بصورت انکد می‌باشد و اسکریپت آزمون با آن داده درست اجرا نمی‌شود. بنبراین نیاز است بصورت دستی مقدار password اصلاح شود. بهمین دلیل با اینکه در فرم login تعداد المانهای اشتباه نگاشت شده صفر است اما میزان خودکارسازی آن ۵۰٪ می‌باشد.

حال می‌خواهیم میزان پوشش مدل را برای WebCalndar محاسبه کنیم. همانطور که در فصل ۳ توضیح داده شد. ابتدا یک مدل ناوبری از سیستم استخراج می‌شود. ابزار به گونه ای طراحی شده است، که هنگام ایجاد اسکریپت آزمون از کاربر پرسیده می‌شود که تا چه سطحی عمل آزمون انجام شود. جدول ۳-۴ میزان پوشش مدل برای برنامه webCalendar را تا سطح ۳ نشان می‌دهد.

تعداد سطح	۳
تعداد صفحات آزمون شده	۳۱
تعداد کل صفحات	۴۲
میزان پوشش مدل	۷۳/۸

جدول ۴-۳ میزان پوشش مدل تا سطح ۳

نرم افزارZen cart

Zen cart یک برنامه خرید الکترونیکی منبع باز به زبان php می‌باشد، که از آدرس http://sourceforge.net قابل دانلود است. دلیل انتخاب آن نصب راحتش می باشد، که هر کسی

شکل ۴-۴ - نمایی کلی از صفحا اصلی نرم افزار zen cart

بدون داشتن امکانات سخت افزاری خاص می‌تواند آنرا نصب نماید.

در قسمت قبل نتایج اجرا بر روی نرم افزار webCalendar توضیح داده شد. مهمترین تفاوت بین نرم افزار zen cart و web calendar ساختار سناریوگرای نرم افزار zen cart می‌باشد. طراحی نرم افزار zen

cart به گونه‌ای است که از سناریوهای مختلفی تشکیل شده است. بعبارت دیگر، کاربران این سایت قالبا یک سناریوی خاص را از ابتدا تا انتها دنبال می‌کنند و سپس از سیستم خارج می‌شوند.

جدول ۴-۴ تعداد المانها و تعداد المانهای اشتباه نگاشت شده را به تفکیک هر فرم نشان می‌دهد. فرم cart quantity به منظور تایید یا اصلاح کاهای سبد خرید طراحی شده است. این فرم برای حالتی بررسی شده است که سه کالا برای خرید در سبد موجود است.

نام فرم	تعداد کل المانها	تعدا المانهای اشتباه نگاشت شده	میزان خودکارسازی
contact us	۳	۰	%۱۰۰
redemption code	۱	۰	%۱۰۰
Discount Coupon	۱	۰	%۱۰۰
multiple_products_cart_quantity	۱۱	۱۰	%۹
Gift Certificates	۶	۵	%۱۷
Advanced Search	۹	۴	%۵۶
cart quantity	۹	۳	%۶۷
Checkout address	۴	۰	%۱۰۰
checkout_payment	۸	۰	%۱۰۰
checkout_confirmation	۱	۰	%۱۰۰
quick_find_header	۱	۰	%۱۰۰
manufacturers_form	۱	۰	%۱۰۰
record_company_form	۱	۰	%۱۰۰
music_genres_form	۱	۰	%۱۰۰
quick_find	۲	۰	%۱۰۰
currencies_form	۱	۰	%۱۰۰
میانگین	۶۰	۲۲	%۶۳

جدول ۴-۴ میزان خودکارسازی عمل نگاشت متغیرها به منابع آنتولوژی به تفکیک فرم

همانطور که درجدول ۴-۴ نشان داده شد، میزان خودکارسازی عمل نگاشت در فرم multiple_products_cart_quantity به میزان قابل ملاحظه ای نسبت به دیگر فرمها پایینتر است. دلیل این تفاوت نام متغیرهایی است که در این فرم استفاده شده است. همانطور که قبلا گفته شده است. عملکرد ابزار پیشنهادی بطور زیادی به نام متغیرها وابسته است. هر چه نام متغیرها با مسما تر انتخاب شود عمل نگاشت بین متغیرها و مفاهیم آنتولوژی بصورت خودکارتر انجام می‌شود.

بدلیل اینکه ابزار پیشنهادی هنگام مواجهه با یک صفحه تمام لینکها و فرمهای آنرا امتحان می‌کند. امکان آزمون یک سناریوی خاص وجود ندارد. نرم افزار zen cart که یک سایت خرید الکترونیکی است، و چندین سناریو وجود دارد، را در نظر بگیرید. در سناریوی خرید و تایید سبد خرید، ابتدا کاربر در سایت جستجو کرده و کالاهای مورد نظر خود را با رنگ دلخواه انتخاب و سپس در صورت لزوم سبد خرید را اصلاح و یا تایید می‌کند. اسکریپتی که بر اساس مدل تولید می‌شود، امکان آزمون گام به گام این سناریو را ندارد. به همین دلیل از روش آزمون مبتنی بر مسیر داده شده استفاده می‌کنیم. برای این منظور مسیر سناریو را بعنوان ورودی به ابزار داده و اسکریپتی برای آزمودن همان سناریوی خاص تولید می‌شود. می‌توان از ابزار selenium برای ضبط کردن مسیر سناریو استفاده کرد.

سناریویی شامل انتخاب ۴ کالای مختلف با متد پرداخت حواله پول[1] را در نظر بگیرید. در سناریوی مذکور ۹ نوع فرم پر می‌شود، که در جدول ۴-۵ نشان داده شده‌اند. جدول ۴-۵ نشان می‌دهد که از ۵ نوع فرم، ۴ تا از آنها کاملا بصورت خودکار پر می‌شوند.

نام فرم	تعداد دفعات استفاده شده	میزان خودکارسازی عمل نگاشت
multiple products cart quantity	٤	%۹

[1] Money order

cart quantity	۱	٪۱۰۰
Checkout address	۱	٪۱۰۰
checkout payment	۱	٪۱۰۰
checkout confirmation	۱	٪۱۰۰

جدول ۴-۵ میزان خودکارسازی نگاشت متغیرها در سناریوی خرید ۴ کالا با متد پرداخت حواله پول به تفکیک فرم

۴-۴ پیشنهادات

با توجه به مطالعات انجام شده، نقاط ضعف زیر به ابزار ارائه شده وارد است که برای کار بیشتر در این زمینه ارائه می‌گردد:

- اگر اطلاعاتی در فرمها وارد شود که در پایگاه داده نباشد و داده با استفاده از محدودیتهای آنتولوژی عمومی تولید شده باشد، در صورتی که در جای دیگری از سایت همان اطلاعات را به گونه ای دیگر از کاربر بخواهد. برای تولید داده از محدودیتهای آنتولوژی عمومی استفاده می‌کند و داده‌ای با استفاده از معیار انتخاب مقادیر مرزی و بصورت تصادفی تولید می‌شود. احتمال اینکه داده تولید شده با داده قبلی یکسان باشد تقریبا غیر ممکن است.
- یکی دیگر از مشکلات ابزار ارائه شده در ارزیابی نتیجه آزمون است. فرمی را در نظر بگیرید که اطلاعات شخصی کاربر را می‌گیرد، اگر در فیلد مربوط به سن فرد به جای عدد، متن وارد کنیم، پس از submit فرم انتظار داریم پیغام خطایی مبنی بر نادرست بودن اطلاعات فیلد سن چاپ شود. اما ابزار فقط کد جواب[1]را چک می‌کند و قادر نیست بفهمد که آیا خطای مناسبی گزارش شده یا خیر. برای درک بهتر فرمی که عمل جستجو را انجام می‌دهد در نظر بگیرید. اگر کلمه ورزش را جستجو کنیم، ابزار نمی‌تواند بفهمد که نتایج بدست آمده از نظر معنا با کلمه مورد جستجو تطابق دارد یا خیر.

[1] Response code

- یکی دیگر از اشکالات در مواجهه با عکسهایی است که باید متنی که در عکس دیده می‌شود در فیلد مورد نظر نوشته شود، و محتوای عکسها در هر بار اجرای صفحه متغیر است. امکان انجام چنین عملی توسط ابزار وجود ندارد.
- در صورتی که باز کردن یک لینک روی دیگر اجراها تاثیر بگذارد. بعنوان مثال در حین آزمون سایت خرید الکترونیکی zen cart، لینک logout کلیک شود، در اینصورت ادامه اجرا با مشکل مواجه می‌شود. این مشکل تا حد زیادی با امکان تولید اسکریپت بر مبنای مسیر، برطرف می‌شود اما بدون استفاده از این امکان احتمال رخداد چنین حالتی بسیار زیاد است.

بیشتر مشکلاتی که در بالا ارائه شده بدلیل خاصیت ماشینی بودن و عدم هوشمندی ابزار می‌باشد. اسناد زیادی در دنیای وب در دسترس همه و مخصوصا انسانهاست. اگر بتوان این اسناد را بصورت کاملتری توضیح داد که برای ابزارها و برنامه‌های نرم‌افزاری قابل خواندن باشد، آنگاه دنیای جدیدی از کاربرد برای وب ایجاد می‌شود.[1] RDFa به منظور تلفیق معنا با XHTML بوجود آمده است.

پیش بینی می‌شود کار بیشتر برای این ابزار در زمینه آزمون برنامه‌های تحت وب مبتنی بر تکنولوژی RDFa، بسیاری از مشکلات ارائه شده فوق را حل کند.

[1] **RDFa** (or Resource Description Framework - in - attributes) is a set of extensions to XHTML that is now a W3C Recommendation

منابع و ماخذ

[And05] Anneliese Andrews, J. O. (2005). Testing Web Applications by Modeling with FSMs. *Software and Systems Modeling, Vol.4 NO.3* , 326-345.

[Bel05] Carlo Bellettini, A. M. (2005). TestUml: user metrics driven web application testing. *2005 ACM Symposium on Applied Computing (SAC'05).* Santa Fe, New Mexico, USA.

[Bel04] Carlo Bellettini, A. M. (2004). WebUml: Reverse Engineering of Web Applications. *2004 ACM Symposium on Applied Computing.* Nicosia, Cyprus.

[Chu05] Huey–Der Chu, John E Dobson and I–Chiang Liu, *FAST: A Framework for Automating Statistics–based Testing*, 2005.

[Cra02] Rick D. Craig, Stefan P. Jaskiel, *Systematic Software Testing*, Artech House Publications, 2002.

[Elb03] Sebastian Elbaumt, S. K. (2003). Improving Web Application Testing With User Session Data. *Proceedings of the 25th International Conference on Software Engineering (ICSE'03)* (pp. 49 - 59). Portland, Oregon: IEEE Computer Society Washington, DC, USA.

[Etm09] Kobra Etminani, M. K. (2009). Transforming Relational Databases to the Corresponding Ontologies. *'Networked Digital Technologies' (NDT 2009).* Ostrava: The Czech Republic.

[Ham04] Paul Hamill, *Unit Test Frameworks*, O'Reilly, 2004.

[LiY07] Li, Y. (2007, June). Master thesis, Model Based Testing approach for web applications. 88 pages.

[Mak07] Mäkinen, M. A. (2007, May 22). Model Based Approach to Software.

[Ric01] Fillipo Ricca, p. T. (2001). Analysis and testing of Web applications. *23rd International Conference on Software Engineering (ICSE `01)*, (pp. 25,34). Toronto,Ontario,Canada.

[Ric04] Fillipo Ricca, p. T. (2004). Analysis, Testing and Re-structuring of Web Applications. *Proceedings of the 20th IEEE International Conference on Software Maintenance (ICSM'04)*, (pp. 474-478).

[Ric01-2] Fillipo Ricca, p. T. (2001). Understanding and Restructuring Web Sites with ReWeb. *IEEE MultiMedia*, (pp. 40 - 51).

[Wan08] Minghui Wang, J. Y. (2008). A Static Analysis Approach for Automatic Generating Test Cases for Web Applications. *International Conference on Computer Science and Software Engineering, (CSSE.2008 vol. 2)*, (pp. 751-754). Wuhan, China.