**دانشگاه تهران**

**پردیس دانشکده‌های فنی**

**دانشکدهٔ مهندسی برق و کامپیوتر**

# طراحی منطق دامنه بر اساس تبادل ناهمگام پیغام

نگارش

**وحید ذوقی شال**

استاد راهنما

**دکتر رامتین خسروی**

**پایان‌نامه برای دریافت درجهٔ کارشناسی ارشد در رشتهٔ**

**مهندسی کامپیوتر - گرایش نرم‌افزار**

شهریور ۱۳۹۱

تقدیم به آنان که در خوشی‌هایم همراهی کردند و در ناخوشی‌هایم صبر؛

پدرم، مادرم و همسر مهربانم

## قدردانی

خدای سبحان را سپاس می‌گویم که به من توان و قوه‌ی ذهنی عطا فرمود تا از عهده‌ی مشکلات انجام این پژوهش برآیم.

در ابتدا لازم می‌دانم از جناب آقای دکتر رامتین خسروی که در انجام این پژوهش افتخار استفاده از راهنمایی ایشان را داشتم، تشکر و قدردانی کنم. مطمئناً این کار بدون کمک‌های همه‌جانبه و بی‌شائبه‌ی ایشان امکان‌پذیر نبود. از اعضای هیئت داوران محترم نیز برای فرصتی که در اختیار من قرار دادند تشکر می‌کنم.

# طراحی منطق دامنه بر اساس تبادل ناهمگام پیغام

## چکیده

آزمون مبتنی بر مدل، به معنی تولید خودکار موارد آزمون از مدل کارکردی و صوری سیستم تحت آزمون به صورت جعبه سیاه، که به عنوان راه‌حلی برای مسئله‌ی تولید و اجرای خودکار آزمون‌ها مطرح شده است، استفاده‌ی روزافزونی درسال‌های اخیر داشته است. ایده‌ی آزمون مبتنی بر مدل، اساساً به هدف آزمون سیستم‌های با رفتار پیچیده (و معمولاً همروند) مطرح شده است. با این حال، در این روش‌ها مدل‌سازی رفتار سیستم توسط نمادگذاری‌های سطح پایین (مانند ماشین‌های گذار) انجام می‌شود. برای مثال، در این روش‌ها مدل‌سازی مقادیر داده‌ای به عنوان پارامترهای ورودی و خروجی سیستم تحت آزمون یا اصلاً امکان‌پذیر نیست و یا منجر به تولید مدل‌های بسیار پیچیده‌ای می‌شود.

از سوی دیگر، روش‌هایی برای آزمون خودکار نرم‌افزار معرفی شده‌اند که هدف آن‌ها تولید موارد آزمون شامل داده است. این روش‌ها به جای توصیف رفتار سیستم توسط ماشین‌های گذار، با استفاده از متن برنامه (به صورت جعبه سفید) امکان توصیف روابط میان مقادیر داده‌ای و چگونگی تولید مقادیر داده‌ای به هدف آزمون را مهیا می‌کند.

بر این اساس، در این پژوهش چهارچوبی یک‌پارچه برای مدل‌سازی همزمان رفتارهای مورد انتظار از سیستم تحت آزمون از یک سو و توصیف داده‌های ورودی و خروجی آن از سوی دیگر ارائه شده است. این چهارچوب برای توصیف این موارد از زبان یوام‌ال استفاده می‌کند. به این ترتیب امکان توصیف سیستم‌هایی که هم از نظر رفتاری و هم از نظر داده‌هایی که مبادله می‌کنند، پیچیده هستند فراهم می‌شود. در راستای طراحی این چهارچوب، هم‌چنین آزمون‌گری پیاده‌سازی شده است که موارد آزمون خود را بر اساس مدل‌های یوام‌ال به صورت خودکار تولید می‌کند.

**واژه‌های کلیدی:** *طراحی منطق دامنه، تبادل ناهمگام پیغام، مدل بازیگر، همروندی*

# فهرست مطالب

# فهرست تصاویر

# فصل ۱

# مقدمه

از دیدگاه مهندسی نیازمندی نرم‌افزار[۱]، مجموعه‌ی نیازمندی‌های یک سیستم نرم‌افزاری را می‌توان در دو دسته‌ی کلی *کارکردی*[۲] و *غیرکارکردی*[۳] طبقه‌بندی نمود [۴]. مطابق این طبقه‌بندی، نیازهای کارکردی عملکرد درست نرم‌افزار را از دید کاربران تعیین می‌کنند. در دید غیرکارکردی نیز به چگونگی عملکرد سیستم و شرایط آن می‌پردازد و در واقع مشخص می‌کند که سیستم با چه کیفیتی در محیط اجرای خود فعالیت خواهد نمود. در ادامه‌ی این متن تنها به بررسی نیازهای کارکردی نرم‌افزار خواهیم پرداخت.

برای اطمینان از صحت پیاده‌سازی نیازمندی‌های نرم‌افزار در متن پیاده‌سازی شده روش‌های مختلفی پیشنهاد شده و مورد استفاده قرار می‌گیرد. برای مثال *آزمون نرم‌افزار*[۴]، دسته‌ای از روش‌ها را برای بازرسی مطابقت[۵] متن برنامه[۶] و نیازهای سیستم معرفی می‌کند. در روش‌های آزمون از اِعمال موارد آزمون[۷] بر سیستم تحت آزمون برای سنجش این

---

[۱] software requirement engineering

[۲] functional

[۳] non-functional

[۴] software testing

[۵] conformance

[۶] source code

[۷] test cases

مطابقت استفاده می‌شود. هر مورد آزمون عبارت است از ترتیبی تعریف شده از عملیات بر روی سیستم و هم‌چنین نتیجه‌ای که در قبال این عملیات از سیستم مورد انتظار است [۵].

یک طبقه‌بندی شناخته شده، روش‌های آزمون نرم‌افزار را به دو دسته‌ی کلی *آزمون کارکردی* و *آزمون ساختاری* تقسیم می‌کند [۶]، اگرچه این تقسیم‌بندی کاملاً استاندارد و همه‌گیر نیست. آزمون کارکردی (که معمولاً با نام آزمون جعبه-سیاه[۸] شناخته می‌شود) بدون اطلاع از نحوه‌ی پیاده‌سازی سیستم انجام شده و تنها به توصیف‌های سطح بالای سیستم متکی است. در مقابل ایده‌ی آزمون کارکردی، آزمون ساختاری قرار دارد (که با نام آزمون جعبه-سفید[۹] نیز شناخته می‌شود). در این ایده، از ساختار داخلی کد و اطلاعات مربوط به نحوه‌ی پیاده‌سازی استفاده شده و تلاش می‌شود تمامی حالت‌هایی که درون متنِ برنامه‌ی پیاده شده است مورد بررسی قرار گیرد.

از جمله‌ی روش‌های جعبه-سیاه شناخته شده، *آزمون مبتنی بر مدل*[۱۰] [۷] است. آزمون مبتنی بر مدل تلاش می‌کند تا با مدل‌سازی رفتار سیستم (که از آن به عنوان توصیف[۱۱] سیستم یاد می‌شود) و تحلیل آن، به طور خودکار به تولید موارد آزمون پرداخته و علاوه بر آن مراحل اجرا و بررسی نتایج آزمون را نیز به صورت خودکار انجام دهد. به این ترتیب، در آزمون مبتنی بر مدل مهم‌ترین فعالیت در طراحی آزمون‌ها، ساختن مدلی از رفتار سیستم و تعیین چگونگی ارتباط آن با سیستم اصلی است. با توجه به خودکار بودن مراحل تولید موارد آزمون، لازم است نمادگذاری‌های به کار رفته برای مدل‌سازی با اتکا بر روش‌های صوری[۱۲] طراحی شوند تا امکان پردازش خودکار آن‌ها فراهم شود.

استفاده از ایده‌ی آزمون مبتنی بر مدل خصوصاً در مورد سیستم‌های با توصیف پیچیده بسیار مفید خواهد بود. در این سیستم‌ها، به دلیل تعدّد عملکردهای سیستم، برای اطمینان از صحت کارکرد آن‌ها باید تعداد زیادی مورد آزمون طراحی نمود. واضح است که در صورت استفاده از روش‌های متداول آزمون، نگه‌داری این مجموعه‌ی آزمون دشوار خواهد بود، زیرا به وجود آمدن تغییراتی در متن برنامه ممکن است تغییرات زیادی در آزمون‌های طراحی شده را طلب کند. در مقابل در روش‌های آزمون مبتنی بر مدل با بروز تغییرات در ساختار و متن برنامه‌ی تحت آزمون تنها کافی است مدل توصیف‌کننده سیستم، تغییر کند، در این صورت موارد آزمون مجدداً با استفاده از این مدل جدید به صورت خودکار تولید خواهند شد. مزیت دیگر استفاده از روش‌های آزمون مبتنی بر مدل، خصوصاً برای سیستم‌های با رفتارهای نسبتاً پیچیده،

---

[۸] black-box testing

[۹] white-box testing

[۱۰] model-based testing

[۱۱] specification

[۱۲] formal methods

اطمینان از فراموش نشدن برخی آزمون‌های مهم است. در روش‌های متداول آزمون، به دلیل دستی بودن فرآیند طراحی آزمون‌ها و نیز تعدد آن‌ها، ممکن است برخی از آن‌ها (به خاطر احتمال بروز خطای انسانی) از قلم بیفتند که این احتمال در روش‌های آزمون مبتنی بر مدل (با توجه به سیستماتیک بودن مراحل تولید موارد آزمون) به طور کلی از بین می‌رود.

بسته به نمادگذاری به کار رفته در مدل‌سازی و نیز روش تولید موارد آزمون از روی این توصیف‌ها، روش‌های متفاوتی برای آزمون مبتنی بر مدل ارائه شده است. در این‌جا بر گونه‌ی خاصی از آزمون مبتنی بر مدل به نام *آی‌اوکو*[۱۳] [۸] تمرکز می‌کنیم. در این روش از نمادگذاری *ماشین‌های گذار*[۱۴] برای توصیف رفتار سیستم استفاده می‌شود.

در گونه‌های امروزی آی‌اوکو، موارد آزمون به صورت در-لحظه[۱۵] تولید می‌شوند. به عبارت دیگر آزمون‌گر مبتنی بر مدل در زمان اجرای آزمون‌ها و با توجه به الگوریتم، یک مورد آزمون تولید و بر روی سیستم اعمال می‌کند. هم‌چنین، آزمون‌گر خروجی‌های سیستم را مشاهده و صحت آن را بررسی می‌کند. یکی از دلایل استفاده از روش‌های در-لحظه برای تولید موارد آزمون، احتمال مشاهده‌ی رفتارهای غیرقطعی[۱۶] از سیستم است، به این معنی که ممکن است سیستم به ازای یک ورودی ثابت امکان تولید چندین خروجی مختلف و مجاز را داشته باشد. به این ترتیب آزمون‌گر می‌تواند در زمان تولید موارد آزمون به طور پویا و بر اساس نحوه‌ی رفتار سیستم نسبت به تولید ورودی مناسب برای سیستم اقدام نماید.

## ۱.۱ انگیزه‌ی پژوهش

روش‌های آزمون مبتنی بر مدل کاستی‌هایی نیز دارند. برای مثال در آی‌اوکو، استفاده از ماشین گذار برای توصیف سیستم تحت آزمون می‌تواند منجر به تولید مدل‌های بسیار پیچیده‌ای شود، زیرا ماشین‌های گذار علی‌رغم قدرت بیان بالا، چنان‌چه خواهیم دید، از نظر توصیفی نمادگذاری سطح پایینی محسوب می‌شوند و بنابراین مدل‌سازی جزئیات سیستم ممکن است حجم زیادی از پیچیدگی را در مدل‌ها به وجود آورد.

یکی از مهم‌ترین عوامل ایجاد کننده‌ی پیچیدگی در مدل‌های مورد استفاده در آی‌اوکو، نیاز به مدل‌سازی مقادیر داده‌ای

---

[۱۳] ioco

[۱۴] transition systems

[۱۵] on-the-fly

[۱۶] non-deterministic

در سیستم‌هایی است که از پارامترهای داده‌ای به همراه ورودی‌ها و خروجی‌های خود استفاده می‌کنند. پیچیدگی مدل‌های تولید شده در صورت وجود پارامترهای داده‌ای، با تعداد پارامترها و مجموعه‌ی مقادیری که هر پارامتر می‌تواند (و یا لازم است) به خود بگیرد به طور مستقیم در ارتباط است. البته برای کاستن از این پیچیدگی، راه‌کارهایی نیز ارائه شده است. برای مثال چنان‌چه خواهیم دید، گسترشی از رابطه‌ی آی‌اوکو با مقادیر داده‌ای ارائه شده است که امکان مدل‌سازی پارامترهای داده‌ای را فراهم می‌سازد.

کاستی‌های روش آی‌اوکو منحصر به این موارد نیست. برای مثال در این روش معیاری برای مقایسه‌ی موارد آزمون مختلف وجود ندارد. بنابراین ممکن است برخی موارد آزمون، بدیهی (مثلاً فقط شامل یک ارسال داده به سیستم) و برخی دیگر بسیار پیچیده (مثلاً شامل ده‌ها و یا صدها رفتار متوالی با سیستم) باشند و یا حتی برخی همدیگر را شامل شوند. مثال دیگر، اتکای روش آی‌اوکو بر تولید آزمون‌ها به صورت کاملاً برخط است. استفاده از چنین روشی همواره مطلوب نیست زیرا ممکن است تولید موارد آزمون (به دلیل بزرگ بودن مدل آن) هزینه‌ی زیادی در بر داشته باشد که تکرار این مسئله احتمالاً مطلوب نیست. البته در این پژوهش به این موارد به طور مستقیم پرداخته نخواهد شد، اما چنان‌چه خواهیم دید حاصل این پژوهش می‌تواند راه را برای رفع چنین مشکلاتی هموار نماید.

## ۲.۱ صورت مسئله

تمرکز اصلی این پژوهش بر آزمون مبتنی بر مدل سیستم‌های وابسته به داده[۱۷] قرار دارد. سیستم‌های وابسته به داده معمولاً حجم زیادی از اطلاعات را با محیط خود مبادله می‌کنند و رفتار آن‌ها وابسته به محاسباتی است که بر روی مقادیر داده‌ای انجام می‌دهند.

چنان‌چه پیش از این نیز گفته شد، در حال حاضر گسترشی از روش آی‌اوکو به هدف مدل‌سازی و آزمون این سیستم‌ها معرفی شده است. با این حال این روش نیز از چند جنبه دچار ضعف است. این پژوهش برای یافتن راه‌کاری جامع برای دو مورد از اساسی‌ترین مشکلات این روش تلاش می‌کند. این دو مورد را می‌توان به ترتیب زیر برشمرد:

- این روش در مورد مقادیر داده‌ای که برای آزمون مورد استفاده قرار می‌گیرند اظهار نظر نمی‌کند. به عبارت دیگر در این روش راه‌کاری برای تعیین این‌که چه مقادیر داده‌ای باید در یک سناریوی رفتاری تولید شده (از روی مدل ماشین‌گذار) به سیستم داده شود، ارائه نشده است. به همین سبب در این روش لازم است تا تمامی مقادیر داده‌ای

[۱۷] data dependent

ممکن بر روی سیستم آزمایش شود که به نظر منطقی نمی‌رسد.

از طرف دیگر مطالعه بر روی ادبیات حوزه‌ی آزمون نرم‌افزار نشان می‌دهد که در گذشته پژوهش‌های مستقلی بر روی روش انتخاب مؤثر داده‌های آزمون انجام شده است و این پژوهش‌ها منجر به ارائه‌ی روش‌های متعددی برای شناسایی و دسته‌بندی مقادیر داده‌ای به اهداف آزمون نرم‌افزار شده است. یکی از این روش‌ها، روش *افراز رده‌ای*[۱۸] است. چنان‌که خواهیم دید این روش به طور سطح بالا نشان می‌دهد که چگونه می‌توان دامنه‌ی مقادیر پارامترهایی که بین سیستم و محیط مبادله می‌شوند را به قسمت‌های مناسبی شکست و از بین آن‌ها تنها برخی از مقادیر را برگزید.

با توجه به این موارد، اولین جزء از صورت مسئله‌ی این پژوهش را می‌توان به صورت استفاده هم‌زمان از روش افراز رده‌ای در کنار آی‌اوکو برای تولید موارد آزمون تعریف نمود. در این قسمت تلاش شده است تا این استفاده‌ی هم‌زمان در قالب یک چهارچوب یک‌پارچه صورت گیرد. به این معنی که در چهارچوب حاصل، این دو روش هم از نظر نحو و نمادگذاری که برای توصیف سیستم مورد استفاده قرار می‌دهند و هم از نظر معنایی به صورت کاملاً یک‌پارچه و بدون درز[۱۹] به نظر آیند.

- گسترش داده‌ای آی‌اوکو پشتیبانی ابزاری چندانی ندارد و در حال حاضر هیچ ابزار آزمون‌گری به طور کامل از آن پشتیبانی نمی‌کند. چهارچوب حاصل برای آزمون مبتنی بر مدل در این پژوهش به عنوان مبنایی برای تولید یک مجموعه‌ی ابزار آزمون‌گر کامل، مورد استفاده قرار گرفته است. برای پیاده‌سازی این ابزار از تجربه‌ی پیاده‌سازی آزمون‌گرهای مبتنی بر مدل و هم‌چنین آزمون‌گرهای دیگری که پشتیبانی گسترده‌ای از آزمون مقادیر داده‌ای می‌کنند استفاده شده است.

ادامه‌ی این متن به طور مشروح به روش رسیدن به این اهداف خواهد پرداخت. با این حال قبل از ورود به بحث، برخی از مهم‌ترین دستاوردهای این پژوهش را به طور فهرست‌وار ارائه می‌کنیم.

## ۳.۱ روش پژوهش

ارزیابی عملی با مطالعه موردی

[۱۸] category partition method

[۱۹] seamless

## ۴.۱ روش ارزیابی

GQM

## ۵.۱ خلاصه‌ی دستاوردهای پژوهش

برخی از دستاوردهای این پژوهش را می‌توان به این ترتیب برشمرد:

- **ارائه‌ی راه‌کاری ابتدایی برای طراحی توصیف‌های سیستم تحت آزمون:** کار طراحی توصیف سیستم (چه برای مدل‌های رفتاری و چه برای مدل‌های داده‌ای) باید با استفاده از دانش به دست آمده در فرآیند تولید نرم‌افزار و هم‌چنین اطلاعات مربوط به نحوه‌ی پیاده‌سازی متن برنامه صورت گیرد. در این پژوهش روشی سطح بالا برای استخراج توصیف‌های سیستم با توجه به این موارد ارائه شده است. این روش امکان تولید همزمان مدل‌های رفتاری و داده‌ها را داراست.

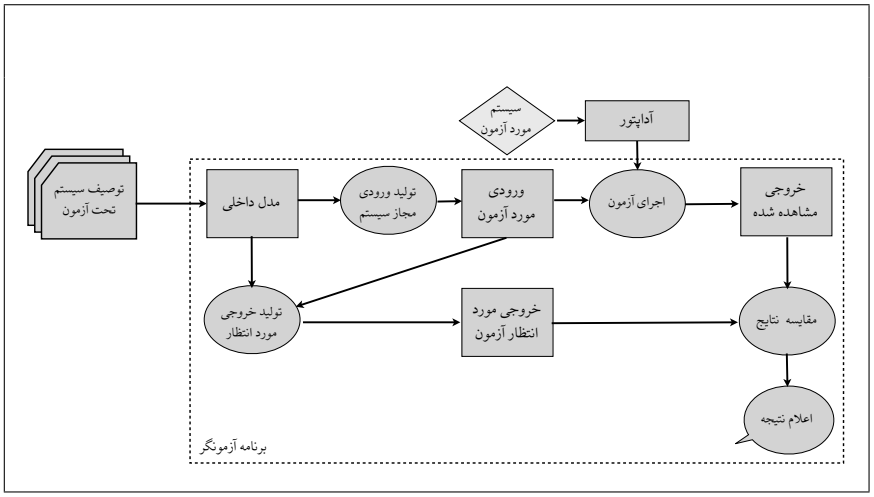
- **ارائه‌ی نمادگذاری یک‌پارچه برای توصیف مدل‌ها:** برای حفظ یک‌پارچگی توصیف‌های رفتاری و داده‌ای، تمامی مدل‌های تولید شده در این چهارچوب در زبان یوام‌ال[۲۰] طراحی شده‌اند. چنان‌چه خواهیم دید در این چهارچوب تلاش شده است تا حد امکان به استانداردهای نحوی و معنایی یوام‌ال وفادار بمانیم.

- **امکان توصیف محدودیت‌های محیط و سناریوهای آزمون:** در روش پیشنهادی این پژوهش، می‌توان یک توصیف ارائه شده از سیستم را چندین بار و در شرایط محیطی متفاوت آزمود. مطابق تعریف، شرایط محیطی در واقع نشان‌دهنده‌ی امکان و یا عدم امکان بروز برخی از رفتارها (که در سیستم پیاده‌سازی شده است) از سیستم است. چنین امکانی مخصوصاً در شرایطی مفید است که سیستم عملکردهای مختلفی داشته باشد و امکان تأثیر این عملکردها بر نحوه‌ی اجرای یک‌دیگر نیز ممکن باشد.

- **نگارش ابزار آزمون:** با توجه به نو بودن بسیاری از ایده‌های این پژوهش، ابزاری جدید به منظور بررسی صحت ایده‌های این کار و هم‌چنین استفاده‌های بعدی، طراحی و پیاده‌سازی شده است. این آزمون‌گر علاوه بر پشتیبانی کامل از آی‌اوکو (و گسترش آن به همراه داده) از تمامی ایده‌های مطرح شده در این پژوهش نیز پشتیبانی می‌کند.

---

[۲۰] Unified Modeling Language (UML)




شکل ۱.۱: نمای سطح بالای مراحل آزمون در روش مبتنی بر مدل [۱]

این ابزار به طور کامل در زبان جاوا[۲۱] پیاده‌سازی شده است. چنان‌چه در فصل ؟؟ خواهیم دید علاوه بر پیاده‌سازی آزمون‌گر، از تعدادی بسته‌ی متن‌باز برای اهدافی مانند طراحی مدل‌ها نیز استفاده شده است.

- **مطالعه‌ی موردی بر روی یک سیستم نمونه:** در نهایت، نمونه‌ی یک سیستم واقعی با استفاده از روش معرفی شده در این پژوهش و هم‌چنین ابزار طراحی شده بر پایه‌ی آن، مورد آزمون قرار گرفته است. از نتایج این مطالعه برای تحلیل کارکرد این چهارچوب در مقابل روش‌های عادی آزمون مبتنی بر مدل (خصوصاً آی‌اوکو) استفاده شده است. به این منظور معیاری عددی (مبتنی بر معیار پوشش متن[۲۲] برنامه) برای مقایسه‌ی آزمون‌های تولید شده توسط روش‌های مختلف تعریف شده است.

مجموعه‌ی دستاوردهای این پژوهش را می‌توان از دید تاثیری که بر ساختاری داخلی برنامه‌ی آزمون‌گر دارند نیز مورد بررسی قرار داد. به این منظور نمای سطح بالا برای تولید آزمون‌ها و بررسی نتایج در روش اصلی آزمون مبتنی برمدل در شکل ۱.۱ آمده است. این ساختار در آزمون‌گر طراحی شده در این پژوهش نیز عیناً وجود دارد با این تفاوت که *اولاً* روش توصیف سیستم تحت آزمون به طور کلی تغییر کرده و *ثانیاً* روش تولید ورودی مجاز سیستم مجدداً تعریف شده است. واضح است که اجزای دیگر آزمون‌گر نیز ممکن است تحت تاثیر این تغییرات، تغییر کنند با این حال موارد ذکر شده مهم‌ترین مواردی است که در این پژوهش مورد طراحی مجدد قرار می‌گیرند.

---

[۲۱] Java

[۲۲] code coverage

## ۶.۱ ساختار پایان‌نامه

برای بررسی این موارد، ساختار این متن در ۶ فصل تنظیم گردیده است:

- فصل ۲ به بررسی برخی پیش‌نیازهای تعریف چهارچوب پیشنهادی می‌پردازد. رابطه‌ی مطابقت آی‌اوکو و روش افراز رده‌ای در این بخش به طور خلاصه مورد بررسی قرار گرفته‌اند.
- در فصل ۳ برخی از کارهای قبلی و مرتبط با مسئله‌ی آزمون مبتنی بر مدل و راه‌حل‌های آن معرفی شده‌اند. هم‌چنین بعضی از مهم‌ترین ابزارهای آزمون نرم‌افزار که مبتنی بر مدل نیستند اما پشتیبانی خوبی برای آزمون سیستم‌های مبتنی بر داده دارند نیز بررسی شده‌اند.
- فصل ۴ ایده‌ی پیشنهادی در این پژوهش به طور مبسوط مورد بحث قرار داده است. این ایده (آن‌چنان که پیش از این نیز اشاره شد) عبارت از ارائه‌ی یک چهارچوب برای آزمون مبتنی بر مدل سیستم‌هایی است که هم از حیث رفتاری و هم از تاثیرپذیری از مقادیر داده‌ای که مبادله می‌کنند، پیچیده‌اند. در این فصل هم به نحو و هم به معنای این چهارچوب پرداخته شده است. برای نحو چهارچوب پیشنهادی از زبان یوام‌ال استفاده شده است و معنای آن نیز بر روش آی‌اوکو (که در فصل ۲ مورد بررسی قرار می‌گیرد) استوار است.
- در فصل ؟؟ یک مطالعه‌ی موردی از آزمون یک سیستم واقعی از حوزه‌ی نرم‌افزارهای مالی، با استفاده از روش پیشنهادی، مورد بررسی قرار خواهد گرفت. برای این منظور معیاری عددی برای مقایسه‌ی آزمون‌های تولید شده توسط این چهارچوب و آزمون‌های تولید شده توسط روش‌های دیگر ارائه شده و نتایج حاصله تحلیل شده‌اند.
- نهایتاً فصل ۶ برخی نکات پایانی را مطرح و در مورد جهت‌گیری‌های احتمالی آینده‌ی این پژوهش مطالبی ارائه می‌کند.

# فصل ۲

# تعاریف اولیه و پیش‌زمینه تحقیق

در این فصل به طور اجمالی بر روش‌های پایه‌ای مورد استفاده در این پژوهش تمرکز خواهیم کرد. در هر مورد تلاش شده است تا تنها از منظر کاربرد در این پژوهش به روش‌ها پرداخته شده و اختصار رعایت شود.

## ۱.۲ مدل بازیگر

آن‌گونه که در فصل قبل اشاره شد، آزمون مبتنی بر مدل روشی برای تولید خودکار موارد آزمون بر اساس مدل‌های صوری از سیستم تحت آزمون است. در این بخش ابتدا نگاهی گذرا به جایگاه آزمون مبتنی بر مدل در میان روش‌های دیگر بازرسی[۱] کارکرد نرم‌افزار انداخته و سپس به نحوه‌ی کار و پایه‌های نظری آی‌اوکو[۲] خواهیم پرداخت. در پژوهش حاضر گسترشی از آی‌اوکو برای آزمون سیستم‌های در حضور داده مورد استفاده قرار گرفته است. با توجه به تعدد نمادهای مورد استفاده و پیچیدگی بیان آن‌ها، در این‌جا ابتدا گونه‌ی اولیه‌ی آی‌اوکو را در بخش ۲.۱.۲ بررسی می‌کنیم و سپس آن‌را در بخش ؟؟ به حضور مقادیر داده‌ای نیز تعمیم می‌دهیم.

---

[۱] inspection

[۲] ioco

### ۱.۱.۲ جایگاه آزمون مبتنی بر مدل در میان روش‌های بازرسی کارکرد نرم‌افزار

روش‌های زیادی برای بازرسی سیستم‌های نرم‌افزاری به هدف اطمینان از کارکرد صحیح آن‌ها ارائه شده است. اگرچه نمی‌توان به طور مشخص در مورد برتری یکی از این روش‌ها به دیگری نظر داد اما می‌توان آن‌ها را از نظر شیوه‌ی استفاده طبقه‌بندی نمود. در زیر، یک طبقه‌بندی برای روش‌های آزمون مبتنی بر مدل، با توجه به نوع محصولات مورد استفاده و روش استفاده از آن‌ها، آمده است.

**روش‌های مبتنی بر بررسی مدل:** در این روش‌ها مدلی از نرم‌افزار مورد بازرسی تحلیل، و درستی مشخصات[۳] مختلفی از آن تحقیق می‌شود. تمامی این روش‌ها در استفاده از مدلی که به طور مجزا طراحی شده است، مشترک می‌باشند. روش‌های مختلفی مانند بررسی مدل[۴] [۹] و یا اثبات قضیه[۵] [۱۰] در این دسته قرار می‌گیرند. مزیت اصلی این روش‌ها حصول اطمینان از درستی خصوصیت مورد بازرسی در مدل سیستم است. دیگر برتری آن‌ها امکان طراحی مدل برنامه به صورت مستقل از پیاده‌سازی و امکان کشف دسته‌ای از خطاها پیش از پیاده‌سازی نهایی برنامه است. علی‌رغم این برتری، در این روش‌ها طراحی مدل‌ها به صورت مجزا ممکن است خود باعث بروز خطاهای جدیدی شود. به علاوه ممکن است برنامه‌ی پیاده‌سازی شده با مدل مورد آزمون قرار گرفته منطبق نباشند. مشکل دیگری که در این روش‌ها باید به آن اشاره کرد هزینه‌ی بالای بازرسی مدل‌ها به دلیل نیاز به ساخت فضاهای حالت به طور کامل و پیمایش آن‌هاست.

**روش‌های بررسی ایستای متن:** [۶] در این روش‌ها خصوصیات مورد نظر مستقیماً روی متن پیاده‌سازی شده‌ی برنامه مورد بازرسی قرار می‌گیرند. در این روش‌ها به جای توصیف مدلی مجزا از سیستم، این مدل‌های به طور خودکار از متن برنامه استخراج می‌شود. قابلیت‌های کنونی این روش‌ها چندان زیاد نیست و در موارد بسیاری هنوز در مرحله‌ی پژوهشی قرار دارد. به این ترتیب باید استفاده از این روش‌ها را به عنوان جایگزینی برای تمامی روش‌های دیگر به دیده‌ی تردید نگریست [۱۱].

**روش‌های مبتنی بر اجرای سیستم:** انواع مختلف آزمون‌ها در این دسته قرار می‌گیرند. این دسته از روش‌ها بر مبنای اجرای برنامه‌ی مورد آزمون با ورودی‌های مشخص و مشاهده‌ی خروجی تولید شده توسط آن در مورد درستی

---

[۳] properties

[۴] model checking

[۵] theorem proving

[۶] static analysis

سیستم اظهار نظر می‌کنند. روش‌های مبتنی بر اجرای سیستم می‌توانند از مراحل ابتدایی پیاده‌سازی تا آخرین مراحل نصب و نگه‌داری نرم‌افزار مورد استفاده قرار گیرند. مهم‌ترین اشکال این روش‌ها نیز عدم تضمین درستی خصوصیت مورد بازرسی حتی بعد از موفقیت‌آمیز بودن همه‌ی مراحل آزمون است.

آزمون مبتنی بر مدل با بهره‌گیری از ایده‌ی مدل‌سازی از روش‌های تحلیل مدل از یک سو و اجرای موارد آزمون تولید شده از سوی دیگر تلاش می‌کند گونه‌ای کارا از آزمون نرم‌افزار را ارائه نماید. آزمون مبتنی بر مدل با استفاده از مدل ارائه شده از رفتار سیستم (که به آن توصیف سیستم[۷] می‌گوییم) و پیمایش فضای حالت، به صورت سیستماتیک موارد آزمونی تولید می‌کند که تمامی حالت‌های مجاز در مدل رفتاری را دربرگیرد. این آزمون‌ها بر روی سیستم تحت آزمون[۸] (که در ادامه‌ی متن حاضر برای راحتی سیستم نامیده خواهد شد) اجرا می‌شوند.

### ؟؟ ۲.۱.۲

رابطه مطابقت ورودی و خروجی[۹][۸] (یا آی‌اوکو) رابطه‌ای است که بر روی دو مدل رفتاری تعریف می‌شود و نشان می‌دهد که آیا رفتار یکی بر دیگری منطبق هست، یا خیر. هم‌چنین آی‌اوکو یک الگوریتم برای ساختن موارد آزمون به طور خودکار از روی توصیف سیستم، مقایسه آن با پیاده‌سازی موجود و اعلام نظر در مورد مطابقت و یا عدم مطابقت این دو ارائه می‌کند.

روش‌های زیادی برای نگارش توصیف سیستم معرفی شده است. برای نمونه می‌توان به نمادگذاری‌های مبتنی بر حالت[۱۰]، نمادگذاری‌های مبتنی بر گذار[۱۱]، نمادگذاری‌ها عملکردی[۱۲] و نمادگذاری‌های تصادفی[۱۳] اشاره نمود[۱]. در این میان آی‌اوکو از نوع خاصی از نمادگذار مبتنی بر مدل به نام *ماشین گذار برچسب‌گذاری‌شده‌ی ورودی و خروجی*[۱۴] (که

---

[۷] system specification

[۸] System Under Test (SUT)

[۹] Input Output Conformance Relation (ioco)

[۱۰] state based notations

[۱۱] transition based notations

[۱۲] operational notations

[۱۳] stochastic notations

[۱۴] Input Output Labelled Transition System (IOLTS)

اختصاراً IOLTS نیز نامیده می‌شود) استفاده می‌کند که در ادامه توضیح داده خواهد شد. در این بخش تلاش شده تا حد امکان وفاداری به نمادگذاری ترتمانس[۱۵] در [۸] حفظ شود. با این حال در مواردی که از نمادگذاری دیگری استفاده شده است، تطبیق کامل آن با نمادگذاری [۸] در پیوست آ انجام شده است.

### ماشین گذار برچسب‌گذاری شده‌ی ورودی و خروجی

**تعریف ۱.۲.** *یک IOLTS عبارت است از یک پنج‌تایی مرتب* $P = \langle S, L_I, L_U, \rightarrow, S_\circ \rangle$ *که در آن*

- $S$ *مجموعه‌ای شمارا و غیر تهی از حالت‌ها؛*
- $L_I$ *مجموعه‌ای شمارا از برچسب‌های کنش‌های ورودی سیستم؛*
- $L_U$ *مجموعه‌ای شمارا از برچسب‌های کنش‌های خروجی سیستم و تعریف می‌کنیم* $L = L_I \cup L_U$ *و* $L_U \cap L_I = \emptyset$*؛*
- $\rightarrow \subseteq S \times (L \cup \{\tau\}) \times S$ *رابطه‌ی گذار را نشان می‌دهد، که در آن برچسب خاص* $\tau$ *نشان‌گر عملیات داخلی سیستم است که نمود بیرونی ندارد؛*
- $S_\circ \subseteq S$ *حالت یا حالت‌های شروع را نشان می‌دهد.*

## ۲.۲ طراحی مبتنی بر دامنه

اگرچه رابطه‌ی معرفی شده از مطابقت ماشین‌های گذار نمادین امکان بازرسی رفتار سیستم به همراه داده را فراهم می‌کند، با این‌حال این روش مشخص نمی‌کند که مقادیر داده‌ای که باید مورد آزمون قرار بگیرند از چه طریقی باید مشخص شوند.

یکی دیگر از جنبه‌های آزمون نرم‌افزار که در کنار روش‌های توصیف رفتار سیستم‌های نرم‌افزاری پیشرفت داده شده است، توصیف نرم‌افزار از حیث دامنه‌ی داده‌های آن است. بر خلاف رابطه‌های مطابقت که از گذار بین حالت‌های مختلف برای توصیف رفتار سیستم استفاده می‌کنند، در این دسته از روش‌ها هر رفتار سیستم معادل با دادن مقداردهی مشخصی به ورودی سیستم در نظر گرفته می‌شود. از آن‌جا که ممکن است مقداردهی‌های مختلفی از مقادیر ورودی

[۱۵] Tretmans

رفتار یکسانی تولید کنند، این مقداردهی‌های داده‌ای بر حسب رفتار سیستم در قبال آن‌ها، به مجموعه‌های هم‌ارز *افراز*[۱۶] می‌شود. هر مجموعه‌ی هم‌ارز از مقداردهی‌های ورودی به سیستم، رفتار یکسانی از سیستم را نیز در پی دارد. به دست آوردن مجموعه‌های هم‌ارز داده‌ای در این روش‌ها باید در یک سناریوی رفتاری مشخص (که از قبل شناخته و معرفی شده است) صورت بگیرد و این روش‌ها فقط به تعیین مقادیر ورودی در هر یک از گام‌های چنین سناریوی شناخته شده‌ای می‌پردازند.

برای مشخص شدن مقادیر داده‌ای در این روش‌ها، از روش‌های مختلفی می‌توان استفاده کرد. این روش‌ها بازه‌ای از بررسی توصیف‌های سطح بالا تا بررسی متن برنامه را پوشش می‌دهد. واضح است که در صورت استفاده از متن پیاده‌سازی شده‌ی برنامه (و یا دانش مستخرج از آن) به عنوان منبع تعریف مقادیر داده‌ای[۱۵، ۱۶]، آزمون تولید شده در حوزه‌ی آزمون‌های ساختاری (یا به عبارت دیگر جعبه سفید) طبقه‌بندی می‌شود.

یکی از روش‌های ساده و در عین حال کارا برای مدل‌سازی داده‌های نرم‌افزار به هدف آزمون روش *افراز رده‌ای*[۱۷] است[۱۷]. این روش علاوه بر آن‌که راه‌کاری برای تولید مجموعه‌های افراز داده‌های ورودی ارائه می‌دهد، چهارچوبی برای تولید موارد آزمون نیز بر اساس آن‌ها معرفی می‌کند.

تولید توصیف‌های سیستم و رده‌های داده‌ای در این روش در چند مرحله انجام می‌شود که در ادامه به طور خلاصه آن‌ها را بررسی می‌کنیم:

۱. اولین مرحله‌ی این روش تحلیل توصیف سیستم و تجزیه‌ی از نظر رفتاری به *واحدهای کارکردی*[۱۸] است. خصوصیت این واحدها این است که هر یک دارای قابلیت آزمون مجزا هستند. رفتار مورد انتظار از هر یک از این واحدهای کارکردی نیز باید شناسایی و ثبت شود. این رفتار عبارت از توالی ورودی‌هایی است که به سیستم داده خواهد شد و خروجی‌هایی باید در قبال آن‌ها دریافت شود.

برای هر یک از واحدهای کارکردی شناسایی شده، لازم است مجموعه‌ای از *پارامترها* نیز معرفی شوند که در واقع متغیرهای داده‌ای هستند که برای آزمون آن واحد کارکردی مورد استفاده قرار می‌گیرند. باید توجه کرد که مقادیر این پارامترها مشخص کننده‌ی رفتار سیستم خواهد بود، بنابراین، انتخاب صحیح آن‌ها اهمیت قابل توجهی در

---

[۱۶] partition

[۱۷] category partition

[۱۸] functional unit

کیفیت نتیجه‌ی آزمون دارد. هر پارامتر توصیف شده را در این روش یک رده[۱۹] می‌نامند.

**مثال.** یک برنامه‌ی مرتب‌سازی[۲۰] را که یک آرایه از اعداد از کاربر دریافت کرده و آن را به صورت مرتب باز می‌گرداند، در نظر بگیرید. با فرض این‌که طول آرایه‌ی ورودی نیز از کاربر دریافت شود و گونه‌های مختلف داده‌ای امکان مرتب شدن با این تابع را داشته باشند، می‌توان رده‌هایی مانند «طول آرایه‌ی ورودی»، «گونه‌ی داده‌ای» و «جایگاه بزرگترین و کوچکترین عنصر در آرایه‌ی ورودی» را برای آزمون چنین سیستمی معرفی نمود.

هم‌چنین توالی رفتاری آزمون را نیز برای واحد کارکردی معرفی شده در این مثال را می‌توان به صورت زیر بیان نمود:

۱. *طول آرایه به سیستم وارد کن.*
۱. *آرایه‌ی مقادیر را به سیستم وارد کن.*
۳. *آرایه‌ی پاسخ را از خروجی سیستم دریافت کن.*

۲. هر رده را می‌توان در مرحله‌ی بعدی به تعدادی گزینه[۲۱] افراز نمود. هر یک از این گزینه‌ها معادل با یک مقدار داده‌ای مشخص است که با توجه به رفتار سیستم مورد آزمون به گونه‌ای انتخاب شده است که رفتاری خاص از سیستم را مورد مشاهده قرار دهند.

در مثال بالا یک نمونه از مقداردهی ممکن آن است که چهار گزینه از طول آرایه را برابر مقادیر {۰، ۱، ۲، ۱۰۰} قرار داده و چهار افراز به صورت *طول آرایه* = ۰، *طول آرایه* = ۱، ۱۰۰ ≤ *طول آرایه* ≤ ۲ و *طول آرایه* > ۱۰۰، تشکیل داد. باید توجه کرد که تعیین این مقادیر خاص تنها بر اساس این اطلاع است که احتمال بروز خطا در این سیستم در قبال تغییر در طول آرایه وجود دارد.

در صورتی که اطلاعی از نحوه‌ی پیاده‌سازی کارکرد مورد آزمون وجود داشته باشد می‌توان آن‌را نیز در فرآیند انتخاب گزینه‌ها دخیل نمود. در مثال بالا در صورتی که بدانیم در ساختار حافظه‌ای که برنامه روی آن اجرا می‌شود، آرایه‌ها را در بلوک‌های ۲۵۶تایی ذخیره می‌کند، می‌توان افرازی به شکل *طول آرایه* < ۲۵۶ ، *طول آرایه* = ۲۵۶ و *طول آرایه* > ۲۵۶ معرفی نمود.

۳. با توجه به خصوصیت هر یک از رده‌ها و ارتباط آن‌ها با یکدیگر، در صورت لزوم محدودیت‌های متقابل رده‌ها در این مرحله تعیین می‌شود. این مسئله حائز اهمیت است که با توجه به توصیف سیستم ممکن است همه‌ی

[۱۹] category

[۲۰] sort

[۲۱] choice

ترکیب‌های تمام گزینه‌ها، به عنوان ورودی مجاز محسوب نشوند. در این صورت از تعریف محدودیت بین رده‌های برای حذف حالت‌های نامطلوب استفاده می‌شود.

بعد از طی این مراحل و تکمیل اطلاعات در مورد مقادیر داده‌ای، در نهایت موارد آزمون نهایی با جایگزینی گزینه‌های انتخاب شده از رده‌های مختلف و اجرای رفتار مشخص شده‌ی آزمون برای هر واحد کارکردی، تولید می‌شود.

؟

اگرچه روش افراز داده‌ای راه‌حلی سطح بالا برای رده‌بندی داده‌های آزمون به دست می‌دهد، در این روش تمامی گزینه‌های انتخاب شده باید در یک سطح بیان شوند و امکان طبقه‌بندی داده‌ای در آن وجود ندارد (به این معنی که نمی‌توان یک رده‌ی انتخاب شده را به رده‌های کوچکتری شکست). گسترشی از روش افراز داده‌ای به نام درخت *طبقه‌بندی*[۲۲] این امکان را فراهم می‌کند که هر کدام از گزینه‌های یک رده، خود مجدداً به تعدادی گزینه‌ی دیگر افراز شود [۲]. تکرار این کار منجر به ایجاد یک درخت از مقادیر داده‌ای می‌شود. نهایتاً در این روش، مقداردهی که برای آزمون مورد استفاده قرار می‌گیرد، از انتخاب مقادیر داده‌ای برگ‌های این درخت انتخاب می‌شود.

**مثال.** یک سیستم بینایی ماشین ساده را فرض کنید که کار شناسایی اشکال هندسی را بر عهده دارد. برای آزمون این سیستم یک رده‌بندی نمونه عبارت خواهد بود از *اندازه*، *رنگ* و *شکل* اجسامی که برای آزمون مورد استفاده قرار می‌گیرند. از طرفی گزینه‌ی اشکال مثلثی را می‌توان به سه گزینه‌ی *مثلث‌های متساوی‌الاضلاع*، *متساوی‌الساقین* و *مختلف‌الاضلاع* شکست. به این ترتیب درختی مشابه شکل ؟؟ به دست می‌آید.

[۲۲] classification tree

# فصل ۳

# کارهای پیشین

در این فصل به ارائه‌ی برخی کارهای پیشین و مرتبط به موضوع این پژوهش خواهیم پرداخت. در مورد هر یک از این موارد به ارتباط آن با بحث جاری، کاربرد و یا نقاط تأثیرگذار آن در موضوع این پژوهش و هم‌چنین ضعف ها و نقایص آن‌ها پرداخته شده است.

با توجه به جنبه‌های مختلف این پژوهش، کارهای مرتبط با هر کدام از این جنبه‌ها نیز به طور مستقل مورد بررسی قرار گرفته است. بر این اساس، ساختار این فصل در سه بخش اصلی تنظیم شده است. در بخش ؟؟ برخی ابزارهای آزمون خودکار نرم‌افزار که بر مبنای نظریه‌ی آی‌اوکو رفتار می‌کنند، بررسی می‌شوند. این بررسی‌ها عمدتاً از دو منظر استاندارد و زبان مدل‌سازی و نحوه‌ی ارتباط آزمون‌گر با سیستم تحت آزمون در هر یک از این ابزارها خواهد بود. بخش ؟؟ برخی از ابزارها و یا روش‌هایی که به طور خاص از داده‌های آزمون پشتیبانی می‌کنند را از نظر خواهد گذراند. در نهایت بخش ؟؟ به توضیح تعدادی از کاربردهای صنعتی استفاده از ابزارهای آزمون مبتنی بر مدل و نتایج به دست آمده از آن خواهد پرداخت.

## ۱.۳ الگوهای همگام سازی

ابزارهای گوناگونی برای آزمون خودکار نرم‌افزار تولید و عرضه شده‌اند که بر پایه‌های مختلف نظری مبتنی هستند. تکیه‌ی ما در این بحث بر روش‌هایی است که از آی‌اوکو به عنوان روش رابطه‌ی مطابقت استفاده می‌کنند.

## ۲.۳ طراحی به روش ارتباط ناهمگام

این بخش به بررسی روش‌ها و ابزارهایی می‌پردازد که داده‌ها را به عنوان ورودی و خروجی سیستم در نظر می‌گیرند و با تولید ترکیب‌های مختلف از مقایر داده‌ای و رد و بدل کردن آن با سیستم به انجام وارسی می‌پردازند.

### ۱.۲.۳ مالتی کور

بسته‌ی ال‌تی‌جی[۱] [۳۰] مجموعه‌ی کاملی از ابزارهای آزمون را معرفی می‌کند که هدف آن انجام کلیه‌ی عملیات آزمون از توصیف کارکرد سیستم تا جزئیات داده‌های آن توسط یک ابزار واحد است. ال‌تی‌جی برای مدل‌سازی جنبه‌های مختلف از سه نمادگذاری مختلف پشتیبانی می‌کند. *نمودارحالت یوام‌ال*، *نمادگذاری بی*[۲] *و نمودار کلاس یوام‌ال*. در این ابزار نمودار حالت بیشتر برای توصیف رفتارهای سیستم مورد استفاده قرار می‌گیرد در حالی که توصیف نمادگذاری بی بیشتر برای نمایش ساختارهای داده‌ای مورد استفاده قرار می‌گیرد.

ضابطه‌ی انتخاب موارد آزمون در ابزار ال‌تی‌جی بر پایه‌ی معیارهای مختلف برای پوشش[۳] طراحی شده است. برای مثال در مورد نمودارهای حالت یوام‌ال این ابزار از ضابطه‌های پوششی مثل «طی کردن تمام گذارها» و «طی کردن تمام حالت‌ها» پشتیبانی می‌کند. هم‌چنین در آن از موارد توصیف‌های بی ضابطه مانند «پوشاندن تمامی حالت‌های شرط‌ها» پشتیبانی می‌شود.

باید توجه کرد که الگوریتم تولید موارد آزمون در ال‌تی‌جی بر رابطه‌های مطابقت (مانند آی‌اوکو) متکی نیست بلکه از روش‌های جستجوی فضای حالت و اجرای نمادین محدود شده[۴] استفاده می‌کند. برای تولید رفتارهای آزمون، آزمون‌گر مسیرهای *مرزی* را در نظر می‌گیرد. به یک مسیر، مرزی اطلاق می‌شود که لااقل یکی از متغیرهای موجود در حالت‌های آن مسیر در حالت کمینه و یا بیشینه قرار داشته باشند. مراحل تولید موارد آزمون در ال‌تی‌جی را می‌توان در سه گام اصلی خلاصه نمود:

۱. تولید حالت‌های مختلف رفتار سیستم با استفاده از مدل رفتاری داده‌شده به سیستم (مانند ماشین‌های حالت)

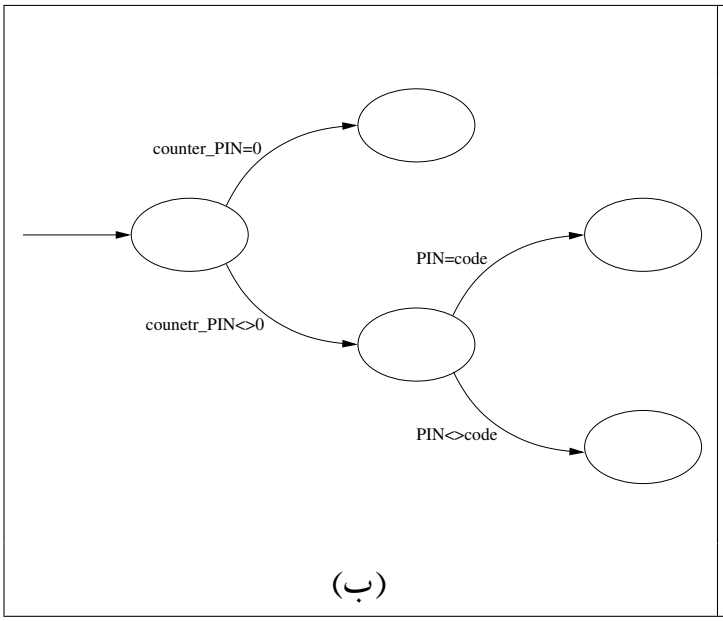
---

[۱] LEIRIOS Test Generator (LTG)

[۲] B Notation

[۳] coverage

[۴] constraint based symbolic execution




(ب)

```
1  sw1, sw2 ← VERIFY PIN(code) =
2      PRE
3          code ∈ {code1,code2,code3}
4      THEN
5          IF (counter P IN = 0)
6          THEN
7              sw1 := 698 ||
8              sw2 := 3
9          ELSE
10             IF (PIN = code)
11             THEN
12                 counter PIN := 3||
13                 sw1 := 900 ||
14                 sw2 := 0
15             ELSE
16                 counter PIN := counter PIN - 1 ||
17                 sw1 := 63C ||
18                 sw2 := counter PIN - 1
19             END
20         END
21     END;
```

(الف)

شکل ۱.۳: (الف) نمونه‌ای از یک توصیف ساختار داده‌ای در نمادگذاری بی و (ب) شمای درختی این توصیف

۲. محاسبه‌ی مقادیر داده‌ای به هدف یافتن حالت‌های مرزی به ازای هر یک از رفتارهای سیستم

۳. تولید موارد آزمون با استفاده از حالت‌های مرزی و معرفی ترتیب‌هایی از عملیات که در شرط‌های مرزی صدق می‌کنند

شکل ۱.۳.(الف) نمونه‌ای از توصیف یک ساختار داده‌ای مورد آزمون در نمادگذاری بی را نشان می‌دهد. این مثال می‌تواند توصیف کننده‌ی مدل درختی از رده‌بندی داده باشد که در بخش ۲.۲ به آن پرداخته شد. شمای این توصیف درختی در شکل ۱.۳.(ب) آمده است.

با وجود گستردگی قابلیت‌های آزمون‌گر التی‌جی، این ابزار محدودیت‌ها و معایب زیادی نیز دارد که در این‌جا به بیان برخی از این موارد می‌پردازیم:

- آزمون‌گر التی‌جی اگرچه از تولید خودکار موارد آزمون از روی مدل رفتاری سیستم پشتیبانی می‌کند، اما الگوریتم تولید موارد آزمون در آن مبتنی بر رابطه‌های مطابقت (مانند آی‌اوکو) نیست. نتیجه‌ی عدم پشتیبانی از آی‌اوکو ایجاد محدودیت‌هایی بر روی روش مدل‌سازی است. برای مثال، مدل‌های ورودی التی‌جی باید همگی متناهی و قطعی باشند در حالی که در آزمون‌گرهایی مانند تورکس چنین محدودیت‌هایی وجود ندارد.
- علی‌رغم پشتیبانی از مقادیر داده‌ای، در التی‌جی امکان تعریف و توصیف گونه‌های داده‌ای وجود ندارد. این امر

باعث می‌شود که محدودیت‌های زیادی در توصیف انواع گوناگون داده به وجود آید.

- برای مدل‌سازی جنبه‌های مختلف سیستم در ال‌تی‌جی نیاز به استفاده از نمادگذاری‌های مختلفی است که این امر هم باعث دشواری در نگارش توصیف‌ها و خوانایی و هم دشواری در نگه‌داری مدل‌ها می‌شود.
- این آزمون‌گر امکان تولید موارد آزمون به صورت در-لحظه را ندارد و موارد آزمون تولید شده باید ابتدا تولید و نگه‌داری شده و سپس برای آزمون سیستم مورد استفاده قرار گیرند.

سلام

# فصل ۴

# روش طراحی پیشنهادی

در فصول گذشته روش آزمون ...

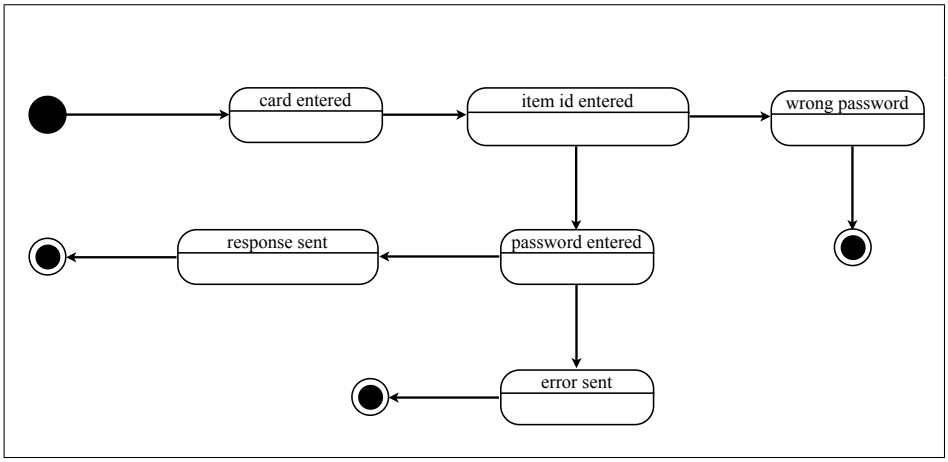
## ۱.۴ معرفی مطالعه‌ی موردی

- خصوصیت ۱
- خصوصیت ۲

### ۱.۱.۴ زیر بخش

در فرآیندهای شناخته شده‌ی تولید نرم‌افزار (مانند RUP) یکی از مهم‌ترین فرآورده‌های تحلیل سیستم، موارد کاربرد[۱] سیستم است. هر مورد کاربرد، در قالب متنی، نشان می‌دهد که رفتار سیستم در تعامل با محیط چگونه است. به عبارت دیگر موارد کاربرد توصیف‌های سطح بالای رفتاری سیستم هستند که به صورت غیر رسمی (ولی با درجات جزئیات متفاوت) بیان می‌شوند. در چهارچوب مورد بحث در این فصل فرض بر آن است که موارد کاربرد برای سیستم مورد نظر تهیه شده و موجود است.

[۱] use cases




شکل ۱.۴: ماشین‌حالت تولید شده برای رفتار پایانه فروشگاهی (سیستم) مثال

### تولید مدل رفتاری سیستم

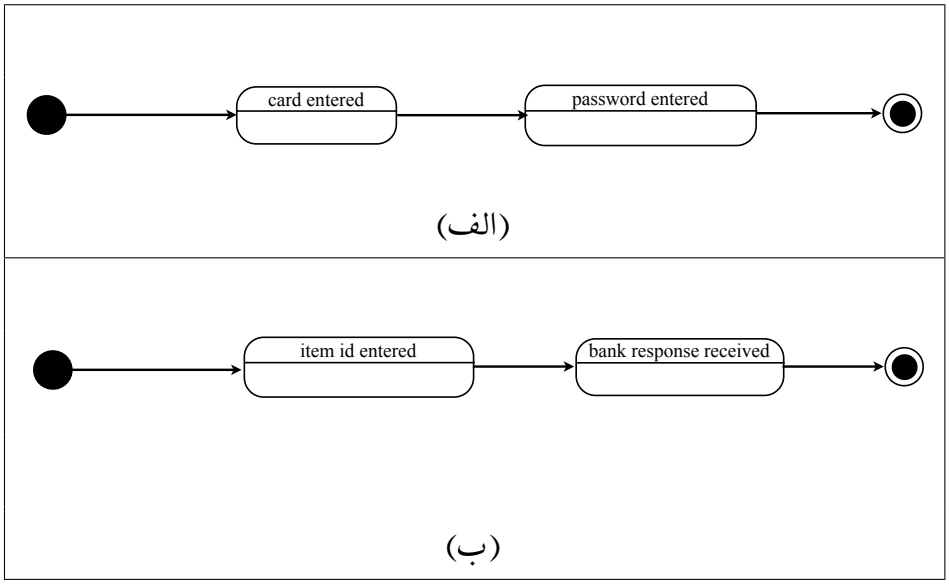
از آن‌جا که رفتار سیستم در قبال هر کنش محیط در مورد کاربرد مشخص شده است، می‌توان از آن مستقیماً برای تولید نمودارهای حالت استفاده نمود. در این تبدیل هر گام در شرح مورد کاربرد که توسط سیستم یا بر روی سیستم اعمال می‌شود، باعث حرکت آن از یک حالت به حالت دیگر خواهد شد. مطابق قرارداد، در ازای هر سناریوی اصلی حتماً یک ماشین مستقل طراحی می‌شود. هم‌چنین در صورتی که مدل‌سازی سناریوهای فرعی در همان نمودار سناریوی اصلی دشوار باشد می‌توان آن‌ها را نیز در ماشین‌های مستقلی مدل‌سازی نمود.

در مدلی که در این مرحله تولید می‌شود، نیازی به مشخص کردن کنش‌هایی که باعث حرکت بین حالت‌های مختلف ماشین می‌شوند وجود ندارد. در واقع نمودار به دست آمده در این مرحله، تنها به هدف مدل‌سازی نمای سطح بالای عملکرد سیستم و مجموعه‌ی حالت‌های آن طراحی می‌شود. مطابق قرارداد، مجموعه‌ی این نمودارهای حالت در یک بسته‌ی[۲] یوام‌ال به نام `def::modeling::system` قرار خواهد گرفت.

**مثال.** نمودار حالت مربوط به سناریوی خرید (که پیش از این مطرح شد) در شکل ۱.۴ رسم شده است. همان‌طور که مشاهده می‌شود، این سناریو از زمانی که یک مشتری کارت خود را به دستگاه نشان می‌دهد شروع شده و با توجه به نحوه‌ی اجرای سناریوهای اصلی و فرعی در یکی از سه نقطه‌ی پایانی متوقف می‌شود. در این نمونه‌ی خاص، با توجه به حجم نسبتاً پایین مدل تولید شده، سناریوهایی فرعی نیز در کنار سناریوی اصلی در یک ماشین مدل‌سازی شده‌اند.

---

[۲] package




شکل ۲.۴: ماشین‌حالت تولید شده برای دو اکتور محیطی در فرآیند خرید. (الف). مشتری و (ب) صندوق‌دار

### تولید مدل رفتاری محیط

آن‌چه که در ادامه‌ی این متن *محیط عملکرد* و یا اختصاراً محیط نامیده می‌شود، به طور دقیق عبارت است از اکتورهایی که در یک مورد کاربرد با سیستم در ارتباطند. همان‌طور که پیش از این نیز گفته شد، برای حفظ هم‌خوانی ماشین‌های طراحی شده با مفهوم نمودار حالت یوام‌ال هر نمودار از رفتار محیط باید شامل عملکرد **فقط** یک اکتور باشد.

**مثال.** نمودارهای محیط برای دو اکتور مشتری و صندوق‌دار مثال مورد کاربرد خرید، در شکل ۲.۴ آمده است. همان‌طور که مشاهده می‌شود در هر نمودار تنها رفتار مربوط به همان اکتور ذکر شده است.

اگرچه در مورد چگونگی تاثیر مدل‌های محیط بر رابطه‌ی مطابقت و الگوریتم تولید موارد آزمون در بخش بعد به تفصیل بحث خواهد شد، اما لازم است در این‌جا به مفهوم نمودارهای محیط توجه شود. توصیف محیط آزمون در واقع محدودکننده‌ی رفتار سیستم محسوب می‌شود. در مثال بالا در صورتی که مشتری هنگام وارد کردن رمز عبور همیشه رمز عبور صحیح را وارد نماید، با توجه به نمودار حالت مشتری سیستم هرگز وارد حالت "wrong password" (در نمودار رفتار سیستم) نمی‌شود، بنابراین این مسیر هرگز اجرا نخواهد شد.

همانند آن‌چه در مورد نمودارهای سیستم ذکر شد، نمودارهای به دست آمده از محیط در این مرحله نیز کنش‌ها را مدل‌سازی نمی‌کنند. این نمودارها در بسته‌ی `def::modeling::environment` قرار خواهند گرفت.

### مشخص کردن و تعریف گونه‌های داده‌ای

اگرچه این بخش به توصیف مدل‌سازی رفتاری سیستم و محیط اختصاص دارد، اما باید توجه کرد که تکمیل مدل‌های رفتاری نمی‌تواند کاملاً مستقل از داده‌هایی که بین اجزای مدل مبادله می‌شوند، انجام شود. همان‌طور که پیش از این در فصل ؟؟ اشاره شد، تعیین رفتار در سیستم مبتنی بر داده، ممکن است مبتنی بر مقادیر داده‌ای آن باشد. برای مثال ممکن است سیستم بر اساس شرط‌هایی که روی متغیرهای خود قرار داده است، بین دو گذار متفاوت که از یک مکان ماشین نمادین خارج می‌شوند یکی را انتخاب نماید. همین رویه در مورد نمودارهای حالت به کار رفته برای توصیف در این چهارچوب وجود دارد. به همین منظور لازم است تا در خلال توصیف رفتاری، برخی خصوصیات پایه‌ای داده‌های سیستم نیز توصیف شوند.

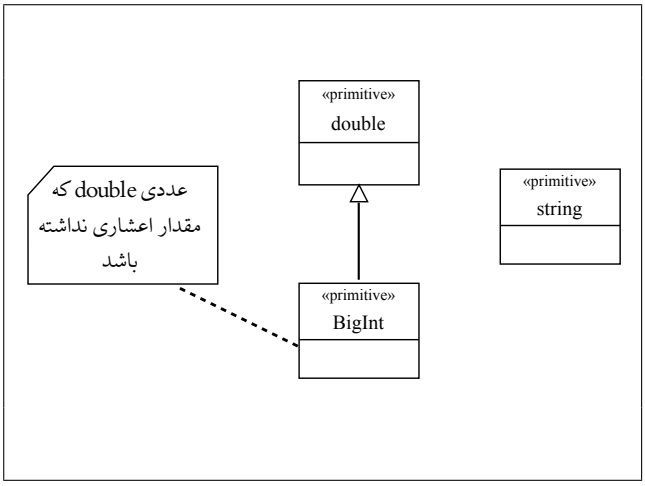
مطابق قرارداد، در چهارچوب پیش‌رو هر مقدار داده‌ای لزوماً دارای گونه‌ای است که باید از پیش مشخص شده باشد. به همین منظور لازم است قبل از افزودن هر مقدار داده‌ای به توصیف‌ها، گونه‌ی آن مقدار تعریف شده باشد. یکی از منابع اصلی برای تشخیص گونه‌های موجود در یک توصیف، همان موارد کاربردی است که پیش از این نیز مورد استفاده قرار گرفته بود. دلیل این امر آن است که در متن موارد کاربرد، معمولاً توصیف داده‌های که در آن مورد کاربرد بین اکتورها رد و بدل می‌شود نهفته است. با استخراج این پارامترها از شرح موارد کاربرد موجود از یک سو و استفاده از دانش تیم پیاده‌سازی برنامه به عنوان یک منبع جانبی، از سوی دیگر می‌توان به مجموعه‌ای از گونه‌های داده‌ای رسید که باید مشخصاً در مدل‌سازی لحاظ شوند.

**مثال.** در مثال فرآیند خرید می‌توان به راحتی پارامترهای داده‌ای زیر را از شرح موارد کاربرد استخراج نمود:

- شناسه‌ی شناسایی کارت مشتری که به سیستم وارد می‌شود.
- شناسه‌ی کالای مورد نظر که توسط صندوق‌دار به سیستم وارد می‌شود.
- شماره‌ی رمز عبور مشتری که توسط خود او به سیستم داده می‌شود.
- نتیجه‌ی خرید که خروجی پایانه‌ی فروش (سیستم) به مشتری و صندق‌دار (محیط) است.

در این مثال فرض می‌شود، اطلاعات خارجی (برای مثال دانش کسب شده از متن برنامه و یا از تیم پیاده‌سازی) نشان می‌دهد که کد شناسایی کارت، کد شناسایی محصول و شماره رمز کارت از نوع عدد طبیعی‌اند (که می‌توانند تا ۱۶ رقم طولانی باشند) و نتیجه‌ی خرید از نوع رشته‌های حرفی[۳] است. هم‌چنین در مورد طراحی کارت فرض می‌شود که رمز

[۳] string




شکل ۳.۴: مجموعه‌ی گونه‌های داده‌ای برای مثال فرآیند خرید

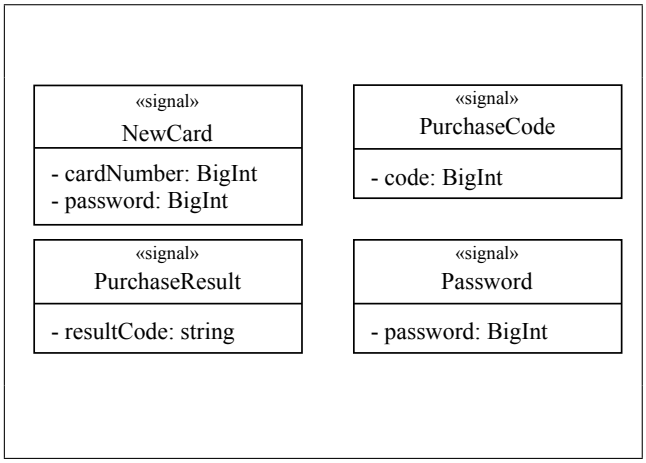
صحیح هر کارت نیز (با همان گونه‌ای که رمز کارت نمایش داده می‌شود) نیز در کارت وجود دارد که در هنگام نشان دادن کارت به دستگاه خوانده می‌شود.

بعد از مشخص شدن گونه‌های داده‌ای، لازم است این گونه‌ها در زبان یوام‌ال تعریف شوند. این گونه‌ها باید همان گونه‌های داده‌ای تعبیه شده در توصیف زبان یوام‌ال یا گسترشی از آن‌ها باشند. در صورتی که یک گونه به صورت گسترشی از یک گونه‌ی پیش‌فرض تعریف می‌شود، باید مجموعه‌ای از محدودیت‌ها را نیز تعریف کند که نشان‌دهنده‌ی تفاوت آن با گونه‌ی اصلی است. در چهارچوب پیشنهادی اجازه‌ی تعریف گونه‌های مستقل (یعنی گونه‌ای که از گونه‌ی دیگری گسترش نیافته است) وجود ندارد. گونه‌های داده‌ای نیز همانند بقیه‌ی اجزای مدل‌سازی این چهارچوب باید با استفاده از کلیشه‌ی[۴] "primitive" از عنصر[۵] کلاس یوام‌ال، تعریف شده و در بسته‌ی `primitives::datatypes` قرارگیرند.

**مثال.** شکل ۳.۴ گونه‌های داده‌ای مثال خرید را نشان می‌دهد. در این شکل گونه‌های string و double گونه‌های تعبیه شده در خود زبان UML هستند و BigInt گونه‌ای گسترش‌یافته از double است، با این محدودیت که اعضای آن نمی‌توانند اعشاری باشد (اما می‌تواند با استفاده از حجم اعداد double اعداد بزرگ را نیز نمایش دهند).

[۴] stereotype

[۵] element




شکل ۴.۴: مجموعه‌ی سیگنال‌ها برای مثال فرآیند خرید

### تعریف سیگنال‌های و پارامترهای آن‌ها

تعریف سیگنال‌ها و پارامترهای آن‌ها با توجه به خروجی مرحله‌ی قبل بسیار آسان است، زیرا اولاً پارامترهای هر سیگنال در مرحله‌ی قبل معرفی شده و ثانیاً گونه‌ی داده‌ای آن نیز تعریف شده است. در نهایت، هر سیگنال ارتباطی لازم است توسط عنصر "signal" که جزئی از ساختار زبان یوام‌ال است، طراحی شده و در بسته‌ی `primitives::signals` قرار گیرد.

**مثال.** شکل ۴.۴ سیگنال‌های طراحی شده برای فرآیند خرید به همراه پارامترهای آن‌ها را نشان می‌دهد.

## ۲.۴ طراحی سیستم به روش ناهمگام

## ۳.۴ الگوها و سبک‌های طراحی

پیش از این، نحو نمادگذاری مربوط به چهارچوب پیشنهادی این پژوهش تشریح شد. در این بخش، در مورد معنای هر یک از اجزای این نمادگذاری بحث خواهد شد. علاوه بر این، مطابقت معنایی این نمادگذاری را با مفاهیم آی‌اوکو مورد بررسی قرار داده و سپس الگوریتم جامعی برای تولید و اجرای یک‌پارچه‌ی آزمون‌هایی که با این نمادگذاری توصیف شده‌اند، ارائه خواهد شد.

### ۱.۳.۴ روشهای coordination

**روش یک**

همان‌طور که پیش از این اشاره شد، در چهارچوب پیشنهادی، توصیف‌های رفتاری چه برای سیستم و چه برای محیط در چندین نمودار حالت بیان می‌شود که هدف از آن کاهش پیچیدگی در طراحی مدل‌هاست. بنا به قرارداد، روش ترکیب این ماشین‌های حالت، روش میان‌گذاری است. به این معنی که

**روش ۲**

همان‌طور که پیش از این اشاره شد، توصیف‌های رفتاری محیط نقش محدودکننده را در تولید موارد آزمون ایفا می‌کند. به عبارت دقیق‌تر با این‌که ممکن است موارد کاربرد مختلفی در یک سیستم پیاده شده باشد، ممکن است محیطی که سیستم در آن فعالیت می‌کند، تنها بخشی از این قابلیت‌ها را مورد استفاده قرار دهد. در این صورت با مدل‌سازی رفتار محیط می‌توان تنها درستی پیاده‌سازی این موارد کاربرد را آزمود. برای نمایش این مفهوم بر اساس نمادگذاری آی‌اوکو، لازم است رابطه‌ی مطابقت را بار دیگر به هدف بررسی مطابقت سیستم در حضور توصیف‌های محیط گسترش دهیم. در گام بعدی الگوریتم تولید موارد آزمون را نیز با توجه به این رابطه‌ی جدید، تعریف خواهیم نمود.

### ۲.۳.۴ سبک های طراحی

سناریوهای آزمون در چهارچوب پیشنهادی، تعیین کننده‌ی رو با توجه به این تعاریف

## ۴.۴ پیاده‌سازی

نحو و معنایی که برای توصیف چهارچوب پیشنهادی در این پژوهش مورد استفاده قرار گرفته است، در قالب یک مجموعه‌ی ابزار نیز پیاده‌سازی نیز شده

# فصل ۵

# ارزیابی

در فصل قبل اجزای چهارچوب پیشنهادی این پژوهش به تفصیل تشریح شد و در م

## ۱.۵ روش ارزیابی

## ۲.۵ ارزیابی کارایی

سبیس

## ۳.۵ ارزیابی تغییرپذیری

سبیس

### ۱.۳.۵ بررسی معیارهای ایستا

با توجه به بزرگی سیستم مورد مطالعه، برای این مطالعه‌ی موردی دو مورد کاربرد از مجموعه‌ی مهم‌تری

### ۲.۳.۵ اعمال تغییرات

**تغییر اول**

## ۴.۵ نتایج ارزیابی

قبل از بررسی نتایج، لازم است برخی نکات در مورد اجرای آزمون‌ها مورد بررسی قرار گیرد. مطابق آن‌چه در فصل قبل بیان شد، برای اجرای متوالی مجموعه‌های آزمون مختلف لازم است معیاری برای خاتمه‌ی هر مجموعه‌ی آزمون معرفی شود. با توجه به یکسان بودن احتمال بروز همه‌ی سناریوهای رفتاری در این مطالعه‌ی موردی و هم‌چنین تجربیات حاصل از چند اجرای آزمون مقدماتی، در این مورد خاص از معیار زمانی برای خاتمه‌ی هر مجموعه و اجرای مجموعه‌ی بعدی استفاده شده است. اگر چه این معیار، ابتدایی‌ترین شرط خاتمه محسوب می‌شود، اما درعوض ساده‌ترین و کم هزینه‌ترین معیار نیز محسوب می‌شود و همان‌طور که اشاره شد در مورد جاری نتایج نسبتاً قابل قبولی نیز تولید می‌کند.

برای ارزیابی آزمون‌ها و مقایسه‌ی حالت‌های مختلف در این مطالعه‌ی موردی، از معیار پوشش متن[۱] استفاده شده است. به این ترتیب مجموعه‌ی آزمونی بهتر فرض می‌شود که پوشش بالاتری از متن برنامه دارد. برای اندازه‌گیری این معیار نیز از ابزار متن باز Cobertura[۳۴] استفاده شده است.

برای ارائه‌ی تحلیلی از کارایی روش معرفی شده در این پژوهش، نتایج آزمون‌های تولید شده توسط مدل‌های بخش قبل (پوشش متن حاصل از انجام آزمون‌های تولید شده بر مبنای این روش) باید با روش شناخته شده‌ی دیگری مقایسه شود. در این مطالعه‌ی موردی این نتایج با نتایج استفاده از روش اولیه‌ی آزمون مبتنی بر مدل مقایسه شده است. یادآوری این نکته ضروری است که الگوریتم آزمون مبتنی بر مدل استاندارد، امکان تبادل داده با سیستم را در طول آزمون ندارد. از آن‌جا که سیستم مورد این مطالعه به طور ذاتی مبتنی بر داده است، غنی کردن آزمون‌های رفتاری تولید شده از روش آزمون مبتنی بر مدل استاندارد، در آداپتور پیاده‌شده برای آن انجام شده است. به این معنی که مقادیر داده‌ای مربوطه پیش از اجرا در آداپتور نوشته شده و به ازای هر پیغام دریافت شده از آزمون‌گر، آداپتور با افزودن این مقادیر به پیغام، آن را کامل کرده و برای سیستم ارسال می‌کند. در مورد پاسخ‌های دریافتی از سیستم نیز مشابه همین روال برای پیغام‌های دریافت شده اتفاق می‌افتد. واضح است که با استفاده از این ایده فقط یک مقداردهی داده‌ای مورد آزمون قرار می‌گیرد.

با استفاده از این روش، آزمون‌های تولید شده بر پایه‌ی روش پیشنهادی این پژوهش با نتایج دو آزمون دیگر مقایسه

---

[۱] code coverage

می‌شوند: یکی آزمونی مبتنی بر آی‌اوکو استاندارد که داده‌های تعبیه شده در آن منجر به انجام کامل عملیات در سیستم (اجرای سناریوی اصلی مورد کاربرد) می‌شود؛ و دیگری آزمونی مبتنی بر آی‌اوکو استاندارد که داده‌های تعبیه شده در آن منجر به بروز خطا و اتمام ناقص عملیات در سیستم (اجرای یکی از سناریوهای فرعی مورد کاربرد) می‌شود. این کار برای هر سه مجموعه‌ی آزمون ذکر شده در بخش قبل انجام شده و نتایج در جدول ؟؟ ذکر شده است.

### ۱.۴.۵ تحلیل نتایج

با داشتن نتای

# فصل ۶

# جمع‌بندی و نکات پایانی

به عنوان جمع بندی متن حاضر، در این فصل به فهرستی از مهم‌ترین دستاوردهای این پژوهش خواهیم پرداخت. در مورد هر یک از این دستاوردها برخی نکات مهم نیز ذکر شده است. بعد از این، برخی از مهم‌ترین کاستی‌های چهارچوب ارائه شده آورده شده است. این کاستی‌ها در هر دو جنبه‌ی نظری و عملی مورد بررسی قرار گرفته‌اند. در نهایت، بر مبنای این موارد برخی جهت‌گیری‌های ممکن برای ادامه‌ی این پژوهش در آینده آورده شده است.

## ۱.۶ دستاوردهای این پژوهش

این پژوهش، چهارچوبی بدیع برای آزمون سیستم‌های نرم‌افزاری بر پایه‌ی روش‌های مبتنی بر مدل ارائه می‌کند که با استفاده از فرآورده‌هایی کاربردی که در فرآیند تولید نرم‌افزار تولید می‌شوند (مانند موارد کاربرد) و هم‌چنین دانش پیاده‌سازی نرم‌افزار برای تولید مدل‌های رفتاری و داده‌ای استفاده کرده، سپس به طور سیستماتیک به تولید خودکار موارد آزمون می‌پردازد. در تعریف این چهارچوب این امکان به وجود آمده که داده‌های آزمون نیز در مدلی هم‌خوان و در یک زبان یکسان با مدل‌های رفتاری (زبان یوام‌ال)، طراحی شده و اطلاعات آن‌ها در مدل‌های رفتاری نیز مورد استفاده قرار گیرد. روش پیشنهاد شده‌ی این پژوهش، در ادامه برای تولید یک ابزار آزمون‌گر مورد استفاده قرار گرفته است. این ابزار از نظر نحوه‌ی تولید موارد آزمون، با توجه به ساختار الگوریتم ؟؟، به صورت در-لحظه عمل می‌کند. در فصل قبل نشان دادیم چگونه از این ابزار برای آزمون یک سیستم واقعی استفاده می‌شود.

در واقع چهارچوب پیشنهاد شده تلاش می‌کند تا مجموعه‌ی به هم‌پیوسته‌ای از فعالیت‌ها برای آزمون را، از اولین مراحل طراحی تا نتیجه‌گیری از مجموعه‌ی آزمون‌ها، پیشنهاد کند. در زیر برخی از مهم‌ترین دستاوردهای هر یک از مراحل این کار آمده است:

- برای طراحی مدل‌ها به طور مستقیم از فرآورده‌های تولید نرم‌افزار مانند مجموعه‌ی موارد کاربرد و یا استانداردهای موجود (مانند استاندارد ISO8583 در مطالعه‌ی موردی) استفاده شده است. هم‌چنین راهکارهایی برای تولید مستقیم مدل‌ها از موارد کاربرد (مثل قواعد مربوط به ایجاد یک ماشین برای هر مورد کاربرد و شکستن آن به اجزا) پیشنهاد گردیده است. به این ترتیب قدمی به سوی مدل‌سازی روش‌مند، به هدف آزمون برداشته شده و از اتکای کامل به دانش ضمنی طراح آزمون پرهیز شده است.
- تمامی توصیف‌ها، چه رفتاری و چه داده‌ای، در زبان یوام‌ال طراحی شده‌اند. علاوه بر تأثیرات مثبتی که استفاده از این زبان در طراحی آزمون‌گر داشت (مانند استفاده از ابزارهای طراحی قدرتمند و مفسّرها)، باید این را نیز افزود که امروزه حجم بسیار وسیعی از افرادی که در حوزه‌ی مهندسی نرم‌افزار فعالیت می‌کنند و حتی برخی از مشتریان سیستم‌های نرم‌افزاری به زبان یوام‌ال تسلط نسبی دارند و یا لااقل با آن آشنا هستند. این نکته خود باعث می‌شود که اولاً استفاده از راه‌کارهای ارائه شده در این پژوهش برای کاربران نهایی آسان شود و ثانیاً ارائه‌ی آن به مشتریان و کاربران خارجی نیز با سهولت بیشتری صورت گیرد.
- الگوریتم آزمون مبتنی بر مدل به همراه داده که در [۱۲] پیشنهاد شده بود از جهات مختلف بهبود داده شده است. یکی از این بهبودها امکان توصیف محدودیت‌های محیطی در کنار توصیف رفتار سیستم است. علاوه بر آن، امکان ایجاد توصیف‌های رفتاری در چند ماشین (چه در مورد توصیف سیستم و چه در مورد رفتار محیط) فراهم آمده است. اگرچه هر دوی این موارد در آزمون‌گرهای دیگری (مانند Uppaal TRON) پیاده‌سازی شده اما هیچ یک بر روی سیستم‌های به همراه داده عمل نمی‌کنند.

  مورد قابل توجه دیگر افزوده شدن شرط خاتمه برای آزمون‌هاست (شرط $\xi$ در الگوریتم ؟؟). این شرط امکان این را فراهم می‌سازد تا در مورد اتمام مراحل آزمون تصمیم‌گیری شود. پیاده‌سازی این شرط نیز با معیارهای متفاوت و با توجه به نیاز صورت خواهد گرفت.
- روشی برای مدل‌سازی داده‌های آزمون معرفی شده است که به طور کامل با الگوریتم تولید موارد آزمون هم‌خوان می‌باشد. این روش با استفاده از روش افراز رده‌ای و با اتکا به فرآورده‌های تولید نرم‌افزار و هم‌چنین دانش حاصل از متن برنامه، می‌تواند مقادیر داده‌ای را به همراه ارتباطات و محدودیت‌های بین آن‌ها مدل‌سازی نماید. استفاده از

زبان مشترک یوام‌ال برای توصیف این موارد تفاوت ماهوی این مدل‌ها را با مدل‌های رفتاری تا حد خوبی پوشش می‌دهد.

- مفهوم سناریوهای آزمون، به هدف مشخص کردن مجموعه‌های آزمون که سیستم را در حضور محدودیت‌های محیطی مختلف مورد آزمون قرار می‌دهند، معرفی گردید. به علاوه، این سناریوها وظیفه‌ی مشخص نمودن مقادیر اولیه‌ی داده‌ای برای سیستم را به ازای هر مجموعه‌ی آزمون بر عهده گرفتند. این مسئله از این بابت حائز اهمیت است که الگوریتم ارائه شده در [۱۲] در مورد تعیین مقادیر اولیه ($\iota$) برای ماشین حالت نمادین، چه در معناشناسی و چه در الگوریتم آزمون، اظهار نظر نمی‌کند اما ترکیب این مقادیر ممکن است بر رفتار سیستم اثری جدی داشته باشد. به این ترتیب استفاده از سناریوهای آزمون می‌توانند راه‌کاری برای این مسئله باشد. در واقع سناریوهای آزمون را می‌توان به عنوان حلقه‌ی اتصال مدل‌های ساختار داده‌ای (که با روش‌هایی مانند افراز رده‌ای شکل می‌گیرند) و مدل‌های رفتاری (که از توصیف‌های رفتاری سیستم مثل موارد آزمون ساخته می‌شوند) دانست.
- آزمون‌گری با پشتیبانی از ایده‌های ذکر شده طراحی و تولید شده است. این آزمون‌گر از توصیف‌های استاندارد XMI (که یک استاندارد مبتنی بر XML برای توصیف نمودارهای یوام‌ال است) بهره گرفته و موارد آزمون مورد نظر را تولید می‌کند. در نهایت، نتیجه‌ی آزمون در قالب یک رأی اعلام می‌شود. با توجه به ضعف پشتیبانی ابزاری از روش آزمون مبتنی بر مدل به همراه داده، طراحی این آزمون‌گر از اهمیت ویژه‌ای برخوردار است.
- کارایی روش پیشنهاد شده با مطالعه‌ی موردی یک سیستم واقعی، سنجیده شده است. اگرچه کارایی این روش را نمی‌توان تنها با استناد به یک مطالعه‌ی موردی نمایش داد، با این حال استفاده از چهارچوب پیشنهادی و ابزارهای آن، تجربیات بسیار ارزنده‌ای در پی داشت که در برخی از مهم‌ترین آن‌ها در فصل قبل بررسی شد. علاوه بر این در انتخاب مورد مطالعه تلاش شد تا نرم‌افزاری جامع در حوزه‌ی نرم‌افزارهای مالی (که خود گستره‌ی بزرگی را شامل می‌شود) انتخاب شود تا به این ترتیب بتوان به طور تقریبی از قابل استفاده بودن این روش در نمونه‌های دیگر نیز اطمینان حاصل نمود.

## ۲.۶ کاستی‌های چهارچوب

چهارچوب پیشنهاد شده در این پژوهش دارای کاستی‌هایی نیز هست که کار بیشتری را می‌طلبد. در این بخش به طور فهرست‌وار به برخی از آن‌ها اشاره می‌کنیم:

- اگرچه تعریف ؟؟ پایه‌ای نظری برای روال تولید موارد آزمون در حضور محدودیت‌های محیطی بنا می‌کند و در ادامه الگوریتم ؟؟ روال تولید موارد آزمون را شرح می‌دهد، با این حال در این مرحله مقایسه‌ای بین قدرت بیان این تعریف از آی‌اوکو با تعریف اصلی آی‌اوکو به همراه داده (تعریف ؟؟) انجام نمی‌شود. این مقایسه لازم است به طور نظری نشان دهد که آیا این دو تعریف، توصیف‌کننده‌ی یک رابطه‌ی مطابقت هستند و یا خیر و در صورت معادل نبودن در مورد جزئیات ارتباط این دو نیز بحث شود.
- در این متن شرط خاتمه‌ی مجموعه‌ی آزمون ($\xi$) به طور کلی تعریف و در مورد برخی از معیارهای تعریف آن نیز بررسی اجمالی انجام شد. با این حال جزئیات تعریف هر یک از این معیارها، مقایسه‌ی آن‌ها از دید معناشناسی با یکدیگر و هم‌چنین روش پیاده‌سازی آن‌ها در این متن پوشانده نشده است. این صورت مسئله خود نیاز به پژوهش‌های مستقلی دارد چنان‌چه برخی از این معیارها هم‌اکنون در قالب پژوهش‌های دیگری در حال توسعه می‌باشند (برای مثال در [۳۵] چهارچوبی برای اندازه‌گیری پوشش مدل ارائه شده است).
- چهارچوب پیشنهاد شده در این متن تنها قادر است به صورت در-لحظه موارد آزمون را تولید کند. این روش اگرچه مزایای بسیاری دارد اما لزوماً در همه‌ی موارد بهترین گزینه نیست. برای مثال در حالت‌هایی که نیاز به آزمون‌های بازگشتی[۱] وجود دارد تولید هرباره‌ی حجم زیادی از موارد آزمون به صرفه نیست. در این موارد روش‌های تولید آزمون به صورت برون‌خط به همراه یک روش برای اولویت‌بندی[۲] موارد آزمون می‌تواند کارساز باشد.
- آزمون‌گر تولید شده، در حال حاضر از قابلیت بررسی گونه‌ها[۳] برای گونه‌های داده‌ای طراحی شده پشتیبانی نمی‌کند. این امر سبب می‌شود تا امکان بروز خطای انسانی در طراحی مقادیر داده‌ای زیاد باشد. این مسئله با توجه به تنوع مقادیر مختلف داده‌ای که می‌تواند مورد استفاده قرار گیرد، بیشتر خودنمایی می‌کند.
- به جز پشتیبانی از داده‌های آزمون، گسترش‌های دیگری از رابطه‌ی مطابقت آی‌اوکو وجود دارد. برای مثال آزمون سیستم‌های زمان‌دار توسط رابطه‌ی مطابقت **rtioco** توصیف شده و در ابزار Uppaal TRON نیز پیاده شده است [۲۴]. برای تکمیل آزمون‌گر پیاده‌شده در این پژوهش لازم است پشتیبانی از این گسترش‌ها نیز به آن افزوده شود.
- تبدیل انجام شده از زبان یوام‌ال به ماشین‌های گذار نمادین هنوز نادقیق است. دلیل این امر، گستردگی بسیار زیاد زبان یوام‌ال و نیاز به تعریف نحو و معنای هر یک از اجزای آن در آزمون‌گر طراحی شده است. در این پژوهش، به

[۱] regression tests

[۲] prioritization

[۳] type checking

طور ضمنی زیرمجموعه‌ی پشتیبانی شده از زبان یوام‌ال معادل عناصری در نظر گرفته شده است که نحو و معنای آن‌ها در این متن تشریح شد.

## ۳.۶ جهت‌گیری‌های پژوهشی آینده

پژوهش حاضر، اگرچه ایده‌ای منسجم را در حوزه‌ی آزمون‌های مبتنی بر مدل دنبال می‌کرد، اما می‌توان آن را به عنوان نقطه‌ی آغازی برای چندین پژوهش مرتبط دانست. برخی ایده‌های مطرح شده در این متن هنوز ابتدایی هستند و برخی نیازها نیز با روش پیشنهادی قابلیت مدل‌سازی را ندارند. اگرچه تکمیل و رفع کاستی‌های ذکر شده در بالا را می‌توان به عنوان کارهای آینده در راستای این پژوهش دانست، اما علاوه بر آن‌ها برخی از جهت‌گیری‌های احتمالی برای پژوهش‌های آینده که از ایده‌ی این پژوهش بهره می‌برند را نیز می‌توان به شکل زیر برشمرد:

- ایده‌ی پیشنهادی در این پژوهش، راه را برای ایجاد یک *فرآیند آزمون نرم‌افزار*[۴] کامل و جامع هموار می‌سازد. چنین فرآیندی لازم است در کنار فرآیند تولید نرم‌افزار قابل استفاده بوده و در آن ارتباط این دو نیز فرآیند نیز تعریف شود.
- ایده‌ی افراز رده‌ای اگرچه چهارچوبی سطح بالا برای مدل‌سازی داده‌های آزمون و ارتباطات آن‌ها فراهم می‌کند. با این‌حال تولید مقادیر داده‌ای باید در این روش به صورت کاملاً دستی انجام شود. در ادامه لازم است روش‌های تولید رده‌ها و مقادیر داده‌ای به صورت نیمه‌خودکار و یا خودکار مورد نظر قرار گیرد.
- بررسی مطالعه‌ی موردی انجام شده در این پژوهش، این ایده را تقویت می‌کند که ممکن است بتوان با استفاده از اطلاعات خاص دامنه‌ی سیستم (در مطالعه‌ی این پژوهش دامنه‌ی سیستم‌های مالی)، به روشی برای تولید توصیف‌های دقیق‌تر و موارد آزمون با کیفیت بالا دست یافت. بنابراین یک مسئله‌ی مهم که می‌تواند به عنوان جهت‌گیری احتمالی این پژوهش مورد نظر قرار گیرد، طراحی روش‌های آزمون و ابزارهایی است که به صورت *خاص- دامنه*[۵] بهینه شده باشند.

---

[۴] software testing process

[۵] domain specific

# پیوست آ

# تطبیق نمادگذاری‌ها

## متن برنامه‌ی طراحی شده به روش ارسال ناهمگام پیغام

سلام

## متن برنامه‌ی طراحی شده به روش شیء‌گرا

تعریف ذکر شده در [۱۲] برای ماشین گذار نمادین به طور ساده به این شکل است:

# کتاب‌نامه


[1] M. Utting, A. Pretschner, and B. Legeard, "A taxonomy of model-based testing," Working paper series 04/2006, University of Waikato, Department of Computer Science. Hamilton, New Zealand, 2006. ج, 7, 11

[2] M. Grochtmann and K. Grimm, "Classification trees for partition testing," *Software Testing, Verification and Reliability*, vol.3, no.2, pp.63–82, 1993. ج, 15

[3] "Standard for financial transaction card originated messages - interchange message specifications – part 1: Messages, data elements and code values," ISO 8583, International Organization for Standardization, 2003. خ

[4] J. Greene and A. Stellman. *Applied software project management*. O'Reilly, 2005. 1

[5] P. Ammann and J. Offutt. *Introduction to software testing*. Cambridge, UK: Cambridge University Press, 2008. 2

[6] P. C. Jorgensen. *Software Testing: A Craftsman's Approach*. Boca Raton, FL, USA: CRC Press, Inc., 1995. 2

[7] L. Apfelbaum and J. Doyle, "Model based testing," in *Software Quality Week Conference*, pp.296–300, 1997. 2

[8] J. Tretmans, "Test generation with inputs, outputs, and quiescence," in *Tools and Algorithms for the Construction and Analysis of Systems* (T. Margaria and B. Steffen, eds. ), vol.1055 of *Lecture Notes in Computer Science*, pp.127–146, Springer, 1996. 3, 11, 12

[9] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, January 1999. 10

[10] D. W. Loveland. *Automated theorem proving: A logical basis*. Elsevier North-Holland, 1978. 10

[11] N. Ayewah, D. Hovemeyer, J. D. Morgenthaler, J. Penix, and W. Pugh, "Using static analysis to find bugs," *IEEE Software*, vol.25, pp.22–29, 2008. 10

[12] A. Gotlieb, B. Botella, and M. Rueher, "Automatic test data generation using constraint solving techniques," *SIGSOFT Software Engineering Notes*, vol.23, no.2, pp.53–62, 1998. 13

[13] N. Tillmann and J. de Halleux, "Pex–white box test generation for .net," in *Tests and Proofs* (B. Beckert and R. Hähnle, eds. ), vol.4966 of *Lecture Notes in Computer Science*, chap. 10, pp.134–153, Springer, 2008. 13

[14] T. J. Ostrand and M. J. Balcer, "The category-partition method for specifying and generating fuctional tests," *Communications of ACM*, vol.31, no.6, pp.676–686, 1988. 13

[15] F. Bouquet, B. Legeard, F. Peureux, and E. Torreborre, "Mastering test generation from smart card software formal models," in *Construction and Analysis of Safe, Secure, and Interoperable Smart Devices, International Workshop* (G. Barthe, L. Burdy, M. Huisman, J.-L. Lanet, and T. Muntean, eds. ), vol.3362 of *Lecture Notes in Computer Science*, pp.70–85, Springer, 2004. 18

[16] "Cobertura project," Available online at: `http://cobertura.sourceforge.net`. 30

[17] L. Frantzen, J. Tretmans, and T. A. Willemse, "Test generation based on symbolic specifications," in *Formal Approaches to Software Testing*, pp.1–15, Springer–Verlag, 2005. 34, 35, 39

[18] L. B. Briones, E. Brinksma, and M. Stoelinga, "A semantic framework for test coverage," in *Automated Technology for Verification and Analysis* (S. Graf and W. Zhang, eds. ), vol.4218 of *Lecture Notes in Computer Science*, pp.399–414, Springer, 2006. 36

[19] K. G. Larsen, M. Mikucionis, and B. Nielsen, "Online testing of real-time systems using uppaal," in *4th Intl. Workshop on Formal Approaches to Testing of Software*, pp.79–94, Springer–Verlag, 2004. 36

[20] G. Babin, P. G. Kropf, and M. Weiss, eds. , *E-Technologies: Innovation in an Open World, 4th International Conference, MCETECH 2009, Ottawa, Canada, May 4-6, 2009. Proceedings*, vol.26 of *Lecture Notes in Business Information Processing*, Springer, 2009. 43

# واژه‌نامه‌ی فارسی به انگلیسی

آزمون بازگشتی ..... regression test
افراز ..... partition
اولویت‌بندی ..... prioritization
آی‌اوکو ..... ioco
بازرسی ..... inspection
بدون درز ..... seamless
بررسی ایستا ..... static analysis
بررسی گونه‌ها ..... type checking
بررسی مدل ..... model checking
برون‌خط ..... offline
بسته ..... package
توصیف ..... specification
توصیف سیستم ..... system specification
خاص-دامنه ..... domain specific
درخت طبقه‌بندی ..... classification tree
در-لحظه ..... on the fly
رده ..... category
روش صوری ..... formal method
سیستم تحت آزمون ..... system under test
عنصر ..... element
کلیشه ..... stereotype
گزینه ..... choice
مرتب‌سازی ..... sort
مشخصه ..... property
مطابقت ..... conformance
مورد آزمون ..... test case
مورد کاربرد ..... use case
واحد کارکردی ..... functional unit
یک‌پارچه ..... integrated

# Integrating Functional and Structural Methods In Model-Based Testing


## Abstract

Model-based testing (i.e. automatic test-case generation based on functional models of the system under test) is now widely in use as a solution to automatic software testing problem. The goal this testing method is to test complex systems (e.g. systems with concurrent behaviors). By the way, it exploits low-level notations (e.g. transition systems) to describe system specifications. Therefore, modeling some aspects of the system, such as input/output data values, may result in high-complexity of the resulted model or it may not possible at all.

On the other hand, there are methods that focus on data-dependent systems. Hereby, they analyze the source code (in a white-box manner), instead if high-level behavioral models, to infer data dependencies and to define test data valuation method. In spite of their power in modeling data items, test behaviors in these methods should be designed manually and therefore, defining numerous and complex behaviors for the test process may lead to difficulties.

In this work, we introduce an integrated framework for modeling both the expected system behaviors and the input/output data structures, consistently. To this end, we have used UML language for modeling purposes. This enables us to describe systems that are both complex in behavior and the data. We have also developed a tool which automatically generates test-cases based on the defined UML models.



**Keywords:** *model-based testing, automatic test generation, test framework, testing data dependent systems, category partitioning methods.*

**University of Tehran**

**School of Electrical and Compuer Engineering**

# Integrating Functional and Structural Methods In Model-Based Testing

**by**

**Hamid Reza Asaadi**

**Under supervision of**

**Dr. Ramtin Khosravi**

**A thesis submitted to the Graduate Studies Office**
**in partial fulfillment of the requirements**
**for the degree of M.Sc**

**in**

**Computer Engineering**

**June 2010**