به نام خدا

گزارش پروژه‌ی کارشناسی

Implementation of Model Checking of Software Product Lines in Presence of Nondeterminism and Probabilities

فهرست

مقدمه

امروزه، خط تولید نرم‌افزار در کاربردهای مختلفی از جمله سیستم‌های امنیتی در جهت درستی‌یابی سیستم‌ها یکی از موارد مورد بحث است. مدل‌سازی و درستی‌یابی صوری خط تولید نرم‌افزار اخیراً به طور گسترده مورد بررسی و پژوهش قرار گرفته‌است. به دلیل امکانِ تعداد بالای محصولات در یک خط تولید نرم‌افزار، درستی‌یابی تک به تک محصولات بسیار هزینه‌بر و حتی غیرعملی است. بنابراین نیاز به روشی برای بررسی همزمان رفتار تمام محصولات (خانواده‌ی محصولات خط تولید) است. مقاله‌ی مورد بحث بر روش probabilistic model checking of software product lines که در آن رفتار هر محصول با استفاده از Markov decision processها نمایش داده می‌شود، تمرکز دارد. در این مقاله با استفاده از یک مدل ریاضی (MDPF)، یک خانواده‌ی محصول را در یک مدل نشان می‌دهد. هم‌چنین یک الگوریتم برای درستی‌یابی ویژگی‌های MDPF که در یک درخت منطق احتمالی_ محاسباتی توصیف می‌شود، آورده شده است.

هدف

به جای درستی‌یابی تک‌تکِ محصولات یک خانواده‌ی محصول، با به دست آوردنِ شباهت‌های موجود در میان محصولات متفاوت در یک سامانه، روشی معرفی می‌شود که درستی کل یک خانواده‌ی محصول توسط یک فرمول PCTL مورد ارزیابی واقع می‌شود. برای این کار، از یک مدل ریاضی به نام Markov Decision Process Family استفاده می‌شود که رفتار کل خانواده را نمایش می‌دهد.

یک MDPF به صورت یک ماتریس نمایش داده می‌شود، و عملیات جمع و ضرب (متفاوت از جمع و ضرب عادی ماتریس‌ها) روی این ماتریس تعریف می‌شود. الگوریتم پیشنهاددادهشده، با توجه به رفتارها و ویژگی‌های مختلف سیستم، و با بررسی زیرمجموعه‌های مختلف محصول، زیرمجموعه‌ای از محصولات که ویژگی خواسته‌شده (توسط فرمول PCTL) را دارد را نتیجه می‌دهد.

هدف از این پروژه پیاده‌سازی این الگوریتم‌ها و بررسی آن توسط ورودی‌های مختلف است.

Software product line:

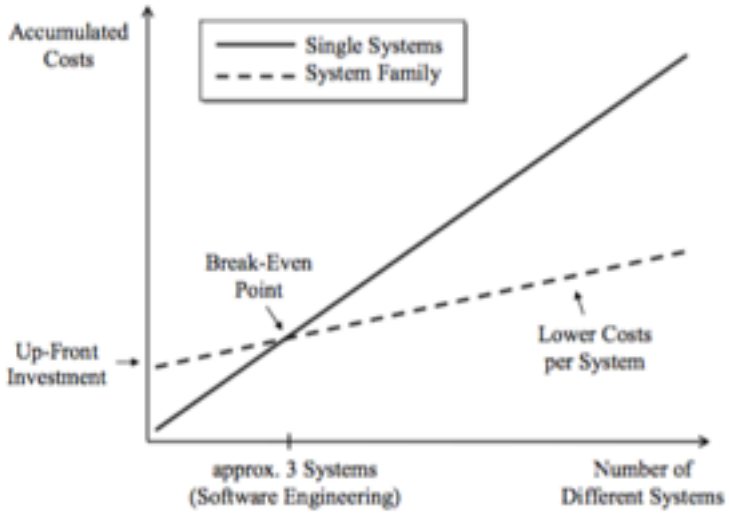
خط تولید نرم‌افزار

با گذشت زمان روش تولید محصولات به طور محسوسی تغییر کرده است. محصولات در گذشته برای هر مشتری تولید می‌شد، در گذر زمان تعداد افرادی که خواهان محصولات متفاوتی بودند افزایش یافت. در صنعت این افزایش تقاضا منجر به اختراع خط تولید در شرکت فورد شد، که امکان تولید انبوه با قیمتی به مراتب ارزان‌تر از تولید تکی محصولات را فراهم می‌کرد. اما خط تولید امکان تنوع در محصولات را کاهش داد.

تقریباً هر دو نوع محصول، تولیدشده به صورت تکی و تولید انبوه در حوزه‌ی نرم‌افزار نیز وجود دارد، که به ترتیب به آن نرم‌افزار شخصی و نرم‌افزار استاندارد گفته می‌شود. به طور کلی، هر دو نوع نرم‌افزار، مشکلات و نواقص خاص خود را دارد، نرم‌افزار شخصی هزینه‌ی نسبی بیشتری دارد در حالی که نرم‌افزار استاندارد تنوع کمی دارد.[1]

دلایل استفاده از خط تولید نرم‌افزار:

۱ ـ کاهش هزینه‌های توسعه‌ی نرم‌افزار:



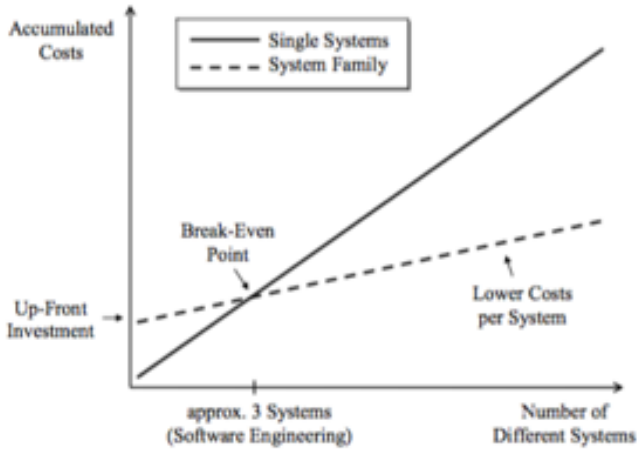
شکل بالا مجموع هزینه را برای تولید n نرم‌افزار شخصی و استاندارد مقایسه می‌کند.

۲ ـ افزایش کیفیت نرم‌افزار:

به دلیل این که نرم‌افزارها در محصولات مختلف بارها مورد تست و ارزیابی قرار می‌گیرند، احتمال کشف خطا و اصلاح آن بسیار بیشتر است، که این کیفیت نرم‌افزار را افزایش می‌دهد.

۳ـ کاهش زمان بازاریابی محصول:

[1] See [Halmans and Pohl 2002] for a treatment of product line engineering for individual vs. mass-market software.



مهندسی خط تولید نرم‌افزار:
مهندسی خط تولید نرم‌افزار، یک الگو برای توسعه‌ی برنامه‌های نرم‌افزاری (سیستم‌های متمرکز بر نرم‌افزار و محصولات نرم‌افزاری) با استفاده از پلت‌فرم‌ها و سفارشی‌سازی انبوه است.

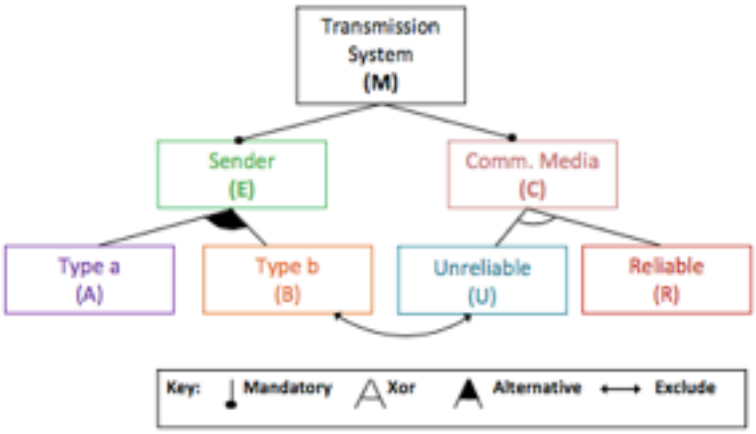
این تعریف هم محصولات کاملاً نرم‌افزاری را شامل می‌شود هم محصولاتی که در آن از نرم‌افزار استفاده می‌شود (نرم‌افزارهای نهفته.)

توسعه‌ی نرم‌افزارها با استفاده از پلت‌فرم‌ها به معناست که هدف از تولید آن استفاده‌ی مجدد از آن به طور گسترده است. ساخت نرم‌افزار برای سفارشی‌سازی انبوه نیز معنای مدیریت تنوع است است، به طوری که تمام شباهت‌ها و تفاوت‌های برنامه‌های مختلف (در حوزه‌ی نیازمندی‌ها، معماری، اجزا، آزمون‌ها) به یک شیوه‌ی مشترک مدل می‌شوند.

مدیریت تنوع، تأثیر شگرفی بر نحوه‌ی توسعه، گسترش و نگه‌داری نرم‌افزار دارد. معمولاً برای افرادی که از نحوه‌ی کار یک نرم‌افزار آگاه هستند، تغییر دادن آن برای یک هدف جدید آسان است. اما این تغییرات، اغلب ساختار اصلی نرم‌افزار را تخریب و کیفیت کد (شامل قابل فهم بودن و قابلیت نگه‌داری) را پایین می‌آورد. برای این‌که این تغییرات به صورت مدیریت‌شده انجام شود، به یک روش تجدیدپذیر (قابل تولید مجدد) نیاز است. مهندسی خط تولید نرم‌افزار با محدود کردن تغییرات ممکن، این روش را فراهم می‌کند.

توصیف یک خط تولید نرم‌افزار (خانواده‌ی محصول):

در یک خط تولید نرم‌افزار، جنبه‌های مختلف محصولات با استفاده از ویژگی‌ها توصیف می‌شوند. ویژگی می‌تواند انواع مختلفی از قبل اجباری یا اختیاری باشد که با روش‌های مختلفی می‌توانند با یکدیگر ارتباط داشته باشند. در بخش II-A مقاله به طور کامل توضیح آن آورده شده است.

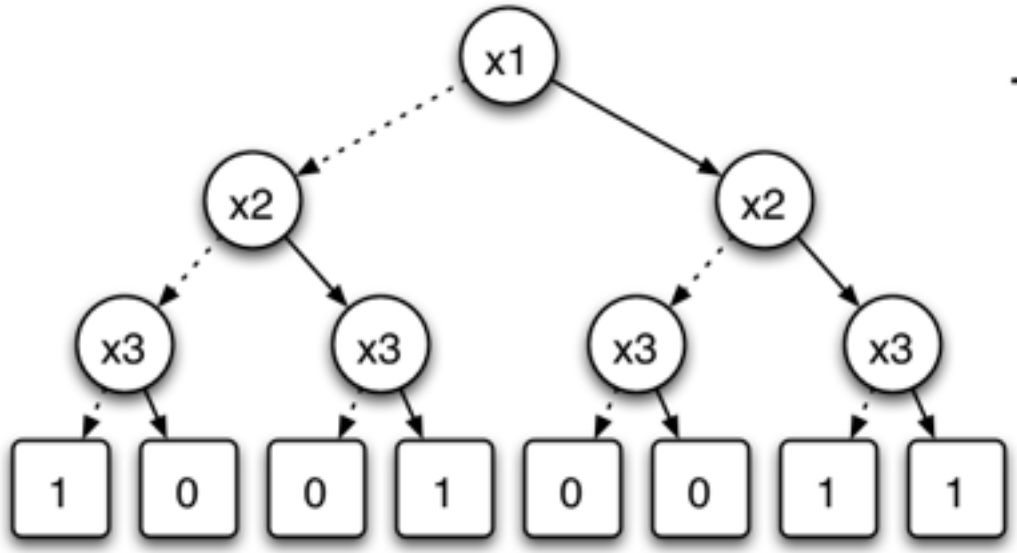

Transmission System (M)
Sender (E)
Comm. Media (C)
Type a (A)
Type b (B)
Unreliable (U)
Reliable (R)
Key:
Mandatory
Xor
Alternative
Exclude

BDD:

Boolean Expression:

عملیات محاسباتی که دو حالت درست (۱) و نادرست (۰) متغیرهای آن را تشکیل می‌دهد و عملگرهای conjunction, disjunction، negation , implication و bi-implication تشکیل می شود، عبارت‌های دودویی را می‌سازند. به این متغیرها متغیرهای گزاره‌ای نیز گفته می‌شود و به این عبارت‌ها منطق گزاره‌ای گویند، گرامر عبارت‌های دودویی به صورت زیر است:

می‌توان یک عبارت دودویی را به صورت یک گراف جهت‌دار بدون دور نمایش داد. به این گراف binary decision digram گفته می‌شود، خط‌چین‌ها حالتی است که متغیر نادرست (۰) باشد و خط پیوسته نشان‌دهنده‌ی درستی عبارت است.



x1	x2	x3	f
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

انجام عملیات روی عبارت‌های دودویی با استفاده از درخت تصمیم دودویی با پیچیدگی زمانی کمتری نسبت به روش‌های دیگر انجام می شود.

از کتاب‌خانه‌ی JavaBDD برای عملیات روی عبارت‌های دودویی استفاده شد.

پیاده‌سازی:

زبان پیاده‌سازی:

پیاده‌سازی این برنامه با استفاده از زبان جاوا انجام شد، که به دلیل وجود کتاب‌خانه‌ی JavaBDD برای انجام عملیات باینری بر روی متغیرها بود.

الگوهای مورد استفاده:

Dependency injection:

یک کلاس برای کار کردن با bddها به نام BDDService پیاده‌سازی شده است که در این کلاس با استفاده از کتاب‌خانه‌ی JavaBDD، اختصاص دادن نام به متغیرها، تبدیل فرمول‌ها به BDD و تمام عملیات‌های دیگر مربوط به BDDها توسط این سرویس انجام می‌شود و با توجه به این که تمام بخش‌های برنامه به این سرویس نیاز دارند به صورت injection کلاس‌های دیگر از این سرویس استفاده می‌کنند.

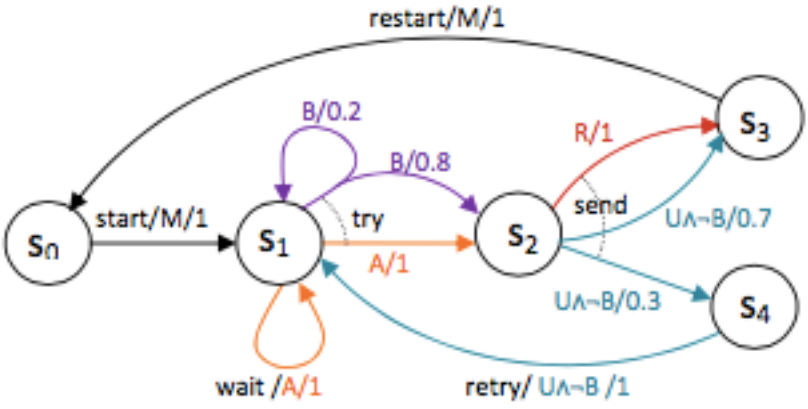
توضیح نحوه‌ی اجرای پروژه:

۱ ـ ورودی برنامه:

ورودی برنامه که یک MDPF است به صورت یک فایل text یا json به برنامه داده می‌شود. فرمت این ورودی به شکل زیر است.

ساختار داده‌ای مورد استفاده برای ذخیره‌ی MDPF:

برای ذخیره‌ی MDPF می‌توان آن را به شکل کامل در یک ماتریس ذخیره کرد و که هر ستون آن یک state را در MDPF نمایش دهد. در روشی که در این پروژه استفاده شده، سعی شده ضمن این که ساختار MDPF به صورت یک state machine حفظ می‌شود تا ساختار برنامه object oriented باقی بماند و برقراری ارتباط در بخش‌های مختلف برنامه با کاربرد آن حفظ شود، ساختار به گونه‌ای طراحی شده که عملیات روی ماتریس‌ها نیز به سادگی بر روی کلاس‌ها قابل پیاده‌سازی باشد.



همان‌طور که در مدل نمونه مشخص است در هر state گذارها با actionها دسته‌بندی می‌شوند برای نگه‌داری گذارها در هر state از یک <Hashmap<action, [Tranistion] استفاده شده است. در نمایش ماتریسی نیز سطرها actionها را نشان می‌دهند و ستون‌ها نشان‌دهنده‌ی stateهاست. که با استفاده از Hashmap دسترسی برای عملیات ماتریس نیز به سادگی و با پیچیدگی زمانی خطی انجام می‌شود.

$$[P_F] = \begin{pmatrix} t/0 & \mathsf{M}/1 & t/0 & t/0 & t/0 \\ t/0 & \mathsf{B}/0.2 & \mathsf{A}/1; \mathsf{B}/0.8 & t/0 & t/0 \\ t/0 & \mathsf{A}/1 & t/0 & t/0 & t/0 \\ t/0 & t/0 & t/0 & \mathsf{R}/1; \mathsf{U} \wedge \neg \mathsf{B}/0.7 & \mathsf{U} \wedge \neg \mathsf{B}/0.3 \\ \mathsf{M}/1 & t/0 & t/0 & t/0 & t/0 \\ t/0 & \mathsf{U} \wedge \neg \mathsf{B}/1 & t/0 & t/0 & t/0 \end{pmatrix}$$

کتاب‌خانه‌ی JavaBDD:

کتاب‌خانه‌ی JavaBDD برای کار کردن با Binary Decision Diagram پیاده‌سازی شده است. این کتاب‌خانه از کتاب‌خانه‌ی Buddy که در زبان c++ پیاده‌سازی شده است استفاده می کند. کتاب‌خانه‌ی Buddy عملیات روی BDDها رو با کارآیی بسیار بالایی پیاده‌سازی کرده است و ویژگی‌های زیر از ویژگی‌های اصلی این کتاب‌خانه است:

dynamic variable reordering,

automated garbage collection,