کارگاه‌های آموزشی مرکز اطلاعات علمی

آموزش مهارت های کاربردی در تدوین و چاپ مقاله

اصول تنظیم قراردادها

مقاله نویسی علوم انسانی

ارائه نمایش مبتنی بر XML از کد برنامه برای تخمین زمان اجرای برنامه‌ها

مهدی سخائی نیا
مربی و کارشناس ارشد نرم افزار
دانشگاه ملایر, دانشکده مهندسی,گروه کامپیوتر
sakhaei@malayeru.ac.ir

سعید پارسا
دانشیار نرم افزار
دانشگاه علم و صنعت ایران ، دانشکده کامپیوتر
parsa@iust.ac.ir

چکیده

تعیین بیشترین زمان اجرای برنامه که اصطلاحاً WCET نامیده می-شود، گام مهم و ضروری در فرایند توسعه و تایید صحت سیستم‌های بی درنگ سخت می‌باشد. یکی از روشهای تحلیل WCET روش تحلیل ایستا می‌باشد. در این روش باید اطلاعات زمانی از کد برنامه استخراج گردد. تعداد تکرار حلقه‌ها, پارامترهای ورودی و خط لوله از عواملی هستند که بر زمان اجرای برنامه تاثیر دارند. برای داشتن ابزاری که تخمین مناسبی از بیشترین زمان اجرای برنامه ارائه دهد, نیاز به در نظر گرفتن و پیاده سازی همه مسائل مطروحه می‌باشد. در این مقاله نمایشی از کد منبع مبتنی بر XML [1] ارائه گردیده که اطلاعات زمانی کد برنامه براحتی در آن درج گردیده و می‌توان بر اساس اطلاعات درج شده در این ساختار زمان اجرای برنامه را محاسبه نمود. همچنین زمینه استفاده از نمایش مبتنی بر XML برای استخراج اطلاعات در پژوهشهای بعدی فراهم گردد.

کلمات کلیدی

بیشترین زمان اجرای برنامه‌ها, تحلیل زمانی, تحلیل ایستا, سیستمهای بی‌درنگ, نمایش مبتنی بر XML.

۱- مقدمه

تعیین کران بالای زمان اجرای برنامه که اصطلاحاً بیشترین زمان اجرای برنامه(WCET) نامیده می شود، گام مهم و ضروری در فرایند توسعه و تایید صحت سیستم های بی درنگ سخت می باشد. متاسفانه بطور معمول امکان محاسبه این کران بالا وجود ندارد. اما با اعمال محدودیتهایی در ساختار برنامه نویسی سیستمهای بی درنگ, مانند عدم استفاده از بازگشتی یا بیان صریح تعداد تکرارحلقه‌ها, می توان بیشترین زمان اجرای برنامه ها را تخمین زد. در روش تحلیل ایستا برای تخمین بیشترین زمان اجرای برنامه، باید طولانی‌ترین مسیر اجرایی برنامه مشخص گردد[۳]. پس از آن بر اساس زمان اجرای هر دستورالعمل که بر روی این مسیر قرار دارد، می‌توان بیشترین زمان اجرای برنامه را تخمین زد. زمان اجرای هر دستورالعمل به کامپایلر و سخت‌افزاری که آنرا اجرا می‌نماید، بستگی دارد. بخصوص در کامپیوترهای مدرن که از امکاناتی مانند خط‌لوله، حافظه cache و یا توازی سطح دستورالعمل بهره می‌برند، زمان اجرای دستورالعمل‌ها می‌تواند متفاوت باشد. برای مشخص نمودن طولانی‌ترین مسیر اجرای برنامه باید پیش‌بینی انشعابها، حدود تکرار حلقه‌ها و عمق فراخوانیهای بازگشتی را بررسی و مشخص نماییم [۴]. استخراج اطلاعات فوق از کد منبع یا کد میانی صورت می‌گیرد. استفاده از یک کد میانی مناسب مطمئناً استخراج اطلاعات را تسهیل می‌نماید.

برای داشتن ابزاری که تخمین مناسبی از بیشترین زمان اجرای برنامه ارائه دهد, نیاز به در نظر گرفتن و پیاده سازی همه مسائل مطروحه می‌باشد. به نظر می رسد پیاده‌سازی همه بخشها مانعی برای دانشجویان و پژوهشگران برای ورود به بحث تحلیل WCET می‌باشد و در نتیجه مانع پیشرفت پژوهش در این حیطه خواهد بود. ملاحظه گروههای تحقیقاتی و همچنین کارگاه تخصصی [2] که در این زمینه هر سال برگزار می‌گردد, موید این موضوع می‌باشد.

در این مقاله قصد داریم نمایشی از کد منبع را ارائه نموده که اطلاعات زمانی کد برنامه براحتی در آن درج گردیده و بتوان بر اساس اطلاعات درج شده در این ساختار زمان اجرای برنامه را محاسبه نمود. همچنین زمینه استفاده از نمایش ارائه شده برای استخراج اطلاعات در آینده فراهم گردد. در ادامه در بخش ۲ به انگیزه کار می‌پردازیم. در بخش ۳ اساس کار تحلیل WCET بیان خواهد. تعیین معیارها برای نمایش مورد نظر در بخش ۴ بیان گردیده است. در بخش ۵ تعریف اجزای ساختار پیشنهادی بیان و در بخش ۶ به کارهای مرتبط پرداخته شده است. ارزیابی در بخش ۷ صورت گرفته و نهایتا نتیجه گیری در بخش ۸ آمده است.

۲- انگیزه

برای بررسی کارایی پژوهشها و مقایسه آن با سایر روشها, باید بخشهای مختلف یک ابزار که تمام مسائل مطرح شده در تخمین ایستای زمان اجرای برنامه را در برداشته باشد, پیاده سازی گردد. علاوه بر آن برای

[1] XML-Base Representation

[2] WCET Workshops

مقایسه روشها و نتایج, نیازمند اجرای برنامه ها بر روی بستر[1] سخت افزاری یکسان می باشد که تحقق این موضوع نیز مشکل می باشد. همچنین پژوهشگرانی که تمایل به کار بر روی فقط یک بخش خاصی داشته باشند, برای مقایسه کارایی و نتایج کار خود دچار مشکل می-باشند مگر اینکه تمام بخشهای لازم را پیاده سازی نمایند.

در صورتیکه بتوان نتایج حاصل از اجرای ابزارها و روشهای متفاوت را با هم تلفیق نمود و یا بخشی از یک روش را تکرار کرد می-توان کیفیت محاسبات و نتایج بدست آمده را افزایش داد. همچنین نیاز است که بررسی صحت درستی هر بخش از ابزار بصورت جداگانه فراهم گشته تا ارزیابی و اشکالزدایی کل روش و ابزار سهلتر گردد. تحقق اهداف فوق نیاز به ابزارهای ماژولار را تبیین می‌نماید که توسط بعضی از محققان مورد توجه و کار گردیده است[۳].

چالشی دیگر اینست که در بعضی از ابزارها, تخمین زمان اجرای برنامه‌ها منجر به شکست گشته و نیاز به اجرای دوباره دارد[۵]. در صورت نگهداری اطلاعات بدست آمده در اجرای قبلی می توان اجرای دوباره را از نقطه شکست ادامه داد که در زمان اجرای ابزار بهبود حاصل می‌گردد.

بعضاً تخمین مناسب بدون استفاده از توضیحات دستی در متن برنامه امکانپذیر نمی باشد. بدین منظور نیاز به زبانی برای درج توضیحات دستی می باشد که در سالهای اخیر چالشها در این زمینه مورد بحث قرار گرفته است[۶].

انگیزه ارائه نمایشی از کد برنامه در قالب xml راه حلی برای مسائل و مشکلات فوق می باشد. ابزارهای تحلیل زمانی می توانند روی این نمایش عمل نموده و اطلاعات زمانی کد برنامه را که بدست آمده است در آن درج نمایند. اگر بخشی از کار توسط روش و ابزاری دیگر انجام گرفت و یا استخراج اطلاعات با یک روش خاص بر روی بستر سخت‌افزاری دیگری صورت گرفت[2], این نمایش امکان بروزرسانی اطلاعات بدست آمده از روش مورد نظر را فراهم آورد. همچنین این نمایش بنحوی ارائه گردد که بتوان زمان اجرای برنامه را با استفاده از تجزیه کننده‌های[3] پیش آماده محاسبه نمود. استقلال از ابزار و بستر اجرایی از اهداف این نمایش است تا براحتی بتوان در برنامه‌های محک[4] برای ارزیابی و مقایسه روشها استفاده گردد. همچنین امکان افزودن توضیحات دستی فراهم گشته که نیاز به ارائه زبانی برای این منظور را منتفی خواهد نمود. در پیاده سازی برخی از روشها از این نوع ساختار نمایشی بصورت ضمنی استفاده شده است[۷], گرچه مشخصاً ساختاری بیان نگردیده و ضرورتهای آن مد نظر قرار نگرفته است.

بعضی از روشها [۸] اطلاعات زمانی مسیرها را ابتدا استخراج کرده و سپس بر اساس آنها محاسبه زمان اجرا صورت می‌پذیرد. برای این نوع روشها یک فرم نمایش صریح اطلاعات مسیرها ضروری می باشد که نمایش پیشنهادی به این امر کمک می‌نماید.

تحلیل WCET وابسته به نوع نمایش کد برنامه بهنگام برنامه-نویسی می‌باشد[۹]. از اینرو با ایجاد سبک نمایش مناسب این امر حصول خواهد شد و تحلیل WCET وابسته به سبک نوشتن کد برنامه-نویس نخواهد بود.

خروجی یک ابزار می‌تواند ورودی ابزار دیگر باشد. عملاً به اشتراک‌گذاری داده‌ها بین ابزارهای مختلف را فراهم می‌کند.

در همه ابزارها بنحوی نتایج حاصله برای استفاده کننده نمایش داده می‌شود[5]. ارائه نمایش مبتنی بر XML سبب می‌گردد خروجی برای تمام ابزارها یکسان بوده و در نتیجه امکان مقایسه نتایج فراهم می‌گردد.

کد بصورت متن- مسطح[6] برای تحلیل نحوی و ساختاری مناسب نمی‌باشد[۱۰]. در حالیکه برای تحلیل WCET نیاز به تحلیل نحوی و معنایی می‌باشد. ارائه ساختاری که بصورت سلسله مراتبی باشد مطمئناً مفید خواهد بود. در این مقاله به بحث نمایشی مناسب برای تحلیل نمی‌پردازیم و بیشتر به ارائه ساختاری که اطلاعات زمانی را نگه داشته و به محاسبه زمان کمک نماید, توجه داریم.

۳- اساس کار تحلیل ایستای WCET

در روش تحلیل ایستا، محاسبه زمان اجرای برنامه در سه بخش جداگانه صورت می‌گیرد که عوامل مختلفی بر آن تاثیر می گذارند [۳]. این سه بخش عبارتند از:

تحلیل جریان[7] : بر اساس کد منبع، میانی ویا object عمل می کند و جریانهای ممکن در برنامه را تعیین می کند مثلاً توالی از دستورات که ممکن است اجرا گردد.

تحلیل سطح پایین[8]: در این بخش تحلیل بر روی کد object و سخت افزار صورت گرفته تا رفتار زمانی برای اجرای دستورات روی سخت افزار مقصد بدست آید. برای پردازنده‌های مدرن که دارای خط لوله[9] و Cache هستند این بخش از اهمیت ویژه ای برخوردارست.

محاسبات[10]: ترکیب نتایج حاصل از تحلیل جریان و سطح پایین می‌باشد که تخمین زمان اجرای برنامه حاصل آن خواهد بود.

۳-۱- تحلیل جریان

هدف از تحلیل جریان تعیین جریانهای ممکن برنامه و پیش‌بینی رفتار پویای برنامه می‌باشد. نتایج حاصل از این بخش اطلاعاتی شامل

1 platform
2 retargetability
3 XML Parser
4 benchmark
5 visualization
6 plain-text
7 Flow analysis
8 Low-level analysis
9 Pipe Line
10 calculation

فراخوانی توابع، تعداد تکرار حلقه‌ها و وابستگی در دستور if می باشد. این تحلیل باید مطمئن و دقیق باشد.

اطلاعات جریان می‌تواند از کد منبع، میانی و ماشین استخراج گردد و حتی ممکن است از اطلاعات جمع آوری شده توسط کامپایلر بدست آید. این بخش خود دارای سه زیر بخش می‌باشد:

- استخراج اطلاعات جریان[1] : بدست آوردن اطلاعات جریان یا با توضیحات دستی یا با روشهای خودکار تحلیل جریان.
- نمایش اطلاعات جریان[2]: نتایج حاصل از بخش قبلی را نمایش دهد و می تواند نتایج حاصل از روشهای مختلف در بخش قبل را ادغام نماید.
- تبدیل برای محاسبات[3]: تبدیل اطلاعات جریان نمایش داده شده به فرمی که برای بخش محاسبات قابل استفاده گردد.

۳-۲- تحلیل سطح پایین

هدف از تحلیل سطح پایین تعیین زمان اجرا برای هر واحد اتمیک از جریان بر اساس ویژگی‌های معماری سخت افزار مقصد می‌باشد. برای بدست آوردن رفتار واقعی برنامه، تحلیل سطح پایین باید بر روی کد object صورت گیرد. این بخش نیز شامل دو زیر بخش می‌باشد:

- تحلیل سطح پایین سراسری[4] : تعیین تاثیر زمانی عوامل وابسته به ماشین که نیاز است بر روی کل برنامه مدلسازی شود. از قبیل ویژگیهایی مانند Cache دستورالعمل، Cache داده، پیش‌بینی انشعاب و بافر ترجمه.
- تحلیل سطح پایین محلی[5] : برای تعیین تاثیر زمانی عوامل وابسته به ماشین که به طور محلی بر روی دستور و دستورات همسایه اعمال می گردد. از قبیل سرعت دستیابی به حافظه و یا همپوشانی حاصل از خط لوله.

۳-۳- محاسبات

هدف از این بخش محاسبه زمان اجرای برنامه بر اساس اطلاعات حاصل از دو بخش قبلی می‌باشد. بطور کلی برای این بخش سه روش مختلف وجود دارد :

- مبتنی بر ساختار درختی[6] . در این روش ساختار اجرای برنامه بصورت یک ساختار درختی در نظر گرفته می‌شود. که هر متد به عنوان گره‌های این ساختار می‌باشند. فرزندان هر گره متدهایی خواهد بود که از داخل این متد فراخوانی می‌گردند. با پیمایش پایین به بالای این درخت، زمان تخمینی بدست می‌آید.
- مبتنی بر مسیر اجرایی[7] . در این روش زمان اجرای برنامه، در مسیرهای مختلف محاسبه می‌گردد. و بر اساس نتایج بدست آمده، طولانی‌ترین مسیر اجرا مشخص شده و بیشترین زمان اجرا تخمین زده می‌شود.
- مبتنی بر تولید ضمنی مسیرهای اجرایی[8]. در این روش محاسبه زمان اجرا بر اساس ساختارهای جبری و منطقی صورت می‌گیرد. برای بخشهای مختلف بصورت بلاکی محاسبات صورت می‌گیرد و بر اساس فرمولهایی(ILP[9]) محاسبات انجام می‌پذیرد.

۴- معیارهای لازم برای ساختار پیشنهادی

ساختار نمایش پیشنهادی علاوه بر امکان نگهداری اطلاعات زمانی کد برنامه, باید محاسبه زمان اجرای برنامه را تسهیل نماید. در این بخش به بیان ساختارهایی در کد برنامه که دارای اطلاعات زمانی هستند و یا به محاسبات کمک خواهند کرد, خواهیم پرداخت. ساختار پیشنهادی باید توانایی نگهداری این اطلاعات را داشته باشد. بر اساس آنچه که در بخش ۳ بیان گردید, بر حسب نوع تحلیل سطح بالا و سطح پایین, ساختارهای درون برنامه اصلی را در دو بخش ۴-۱ و ۴-۲ بررسی خواهیم کرد. پس از آن با توجه نیازمندیهای مطرح شده, در بخش ۴-۳ به بیان دلایل استفاده از XML پرداخته خواهد شد.

۴-۱- ساختارهای موثر در تحلیل سطح بالا

پارامترهای ورودی: داده‌هایی که به عنوان پارامتر ورودی در بخش‌های مختلف برنامه دریافت می‌گردد، بر مسیر اجرای برنامه و بر زمان اجرای آن تاثیر دارد. پارامترهای ورودی بطور کلی به دو دسته تقسیم می‌شوند: مقادیر خارجی که از خارج برنامه وارد می‌شوند. محدودیت موجود در این بخش فقط با توضیحات دستی در کد قابل حل می‌باشد زیرا این داده‌ها توسط دنیای خارج برنامه تولید می‌شوند و مقادیر واقعی آنها فقط در زمان اجرای برنامه مشخص است. نوع دیگر مقادیر داخلی هستند که بوسیله پردازش و محاسبه مقدار متغیرهای برنامه بر اساس تحلیل ایستا، مشخص می‌گردند [۵].

مشخص نمودن حد تکرار حلقه و فراخوانی بازگشتی: حلقه‌ها یکی از مهمترین مسائل در تخمین زمان اجرای برنامه است. مشکل اینجاست که همیشه تعیین حدی برای تکرار حلقه‌ها امکان‌پذیر نمی‌باشد. البته در صورتیکه اشتباهی صورت نگیرد، می‌توان با درج حد تکرار در کد برنامه به عنوان توضیح، این مشکل را حل کرد. عین این مشکل در خصوص تعداد فراخوانیهای بازگشتی وجود دارد. البته سعی بر این بوده که این حدود بصورت خودکار محاسبه گردد [۵] [۱۱].

---

[1] Flow extraction
[2] Flow representation
[3] Calculation conversion
[4] Global low-level analysis
[5] Local low-level analysis
[6] tree-base
[7] Path-base
[8] IPET(Implicit Path Enumeration Techniquel)
[9] Integer Liner Programmin

مسیرهای اجرا نشدنی[1]: در یک برنامه مسیرهای گوناگونی وجود دارد. برای مثال در ۲۰ دستور If، ۲۲۰ یعنی ۱۰۴۸۵۷۶ مسیر اجرایی متفاوت وجود خواهد داشت. در روشهای موجود زمانهای لازم برای اجرای بلاکهای پایه با هم جمع شده و در نتیجه بعضی از بلاکها که هرگز اجرا نمی شوند، به حساب نمی آیند [۱۲]. به مثال شکل (۱) توجه نمایید.

```
[a,b]
a:=top
a>10    if True[ b:=1;]        e1
        if False[ b:=10;]      e2
a>5     if True[ b:=b+3;]      e3
        if False[b:= b+1;]     e4
```

شکل(۱): مثال برای حذف مسیرهای اجرا نشدنی

در مثال شکل (۱) در صورتیکه a>10 مسیر e1;e4 یک مسیر اجرا نشدنی است. یعنی اگر شرط a>10 صحیح باشد، بخش else یعنی a>5 رخ نخواهد داد. حال اگر زمان اجرای بخش then در دستور شرطی a>10 یک واحد زمان بوده و بخش else یک واحد زمان ببرد و در دستور شرطی دوم این مقادیر به ترتیب یک و ۱۰ باشد، بیشترین زمان اجرای تخمینی ۱۰+۱۰ یعنی ۲۰ می باشد. در صورتیکه اگر a>10 باشد، مسیری که سبب بزرگترین زمان اجرا شده هرگز اجرا نمی‌شود و تخمین صحیح ۱+۱۰ یعنی ۱۱ می باشد. شناسایی این مسیرها در شناسایی طولانی‌ترین مسیر و در نتیجه تخمین بیشترین زمان اجرا تاثیر دارد [۵] [۱۲] [۴]. مسیرهای اجرا نشدنی، جدا و متفاوت از کده مرده و کد غیر قابل دسترس می‌باشد [۵].

دستورات شرطی و انتخابی: در دستورات شرطی بر حسب نتیجه عبارت شرطی یکی از دو مسیر اجرایی if-then و if-else اجرا خواهد شد. قاعدتاً زمان اجرای کل بلاک بسته به انتخاب مسیر اجرایی خواهد داشت. در دستورات انتخابی(switch-case) نیز زمان اجرای کل دستور بسته به انتخاب صورت گرفته متفاوت خواهد بود.

۴-۲- ساختارهای موثر در تحلیل سطح پایین

همانطور که دربخش ۳ بیان گردید, باید زمان هر دستور بر روی مسیر اجرایی موردنظر مشخص گردیده تا کل زمان اجرای آن مسیر بدست آید. این زمان دقیقاً مجموع زمان دستورات روی این مسیر اجرایی نخواهد بود. زیرا که امکانات موجود در معماری پردازنده‌های مدرن، سبب می‌شود بعضاً دستورات همزمان اجرا شده و زمان اجرای آنها همپوشانی داشته باشد. بخش سطح پایین از تحلیل زمان اجرا وابسته به این امکانات ریز پردازنده می باشد [۱۲].

زمان اجرای هر دستور: هر دستور در زبان برنامه‌نویسی بر حسب نوع ماشینی که روی آن اجرا می‌گردد, دارای یک زمان اجرا مطلق و مستقل از سایر دستورات می‌باشد. این زمان برای هر دستور باید مشخص گردد.

خط لوله: یکی لز خصیصه‌های پردازنده‌های مدرن خط لوله است. یک خط لوله اجرای دستورات را بوسیله اجرای همزمان دستورات در مراحل جداگانه تسریع می‌نماید. این مراحل معمولا آوردن دستورالعمل، بازگشایی کد، اجرا و نوشتن نتایج می‌باشد.

اجرای تخمینی[2]: اجرای تخمینی وابسته به خط لوله می باشد. یکی از روشهای تسریع، پیش‌بینی جریان دستورالعملها می‌باشد. در این روش سعی می‌گردد دستورالعمل بعدی که اجرا خواهد شد را تخمین و حدس بزنند. در هر قطعه از کد، دستورات انشعاب زیادی وجود دارد. تعیین دستورالعملی که بعد از این انشعاب اجرا می گردد هدف است. همیشه این دستور ممکن است یک دستور پرش محض نباشد و وابسته به دستور قبل از انشعاب باشد که هنوز کامل نشده است. مشخص شدن انشعاب مانند مخاطره داده‌ای، در خط لوله ایجاد وقفه می نماید تا دستورالعمل مناسب به خط لوله وارد شود که به آن مخاطره کنترلی گویند. این مشکل با پیش‌بینی انشعاب قابل حل می‌باشد. البته فقط درصدی از پیش بینی‌ها درست می‌باشد [۵].

کَش(cache) : یکی دیگر از خصیصه‌های پزدازنده‌های مدرن حافظه کَش می‌باشد. حافظه اصلی در یک سیستم کامپیوتری، RAM پویاست که در مقایسه با سرعت پردازنده خیلی کند می‌باشد. سرعت حافظه کَش که بصورت ایستا می‌باشد، چندین برابر حافظه RAM می‌باشد. کَشها روی پردازنده‌های مدرن معمولا به دو بخش تقسیم می گردند: ناحیه حافظه برای داده‌ها و ناحیه‌ای برای دستورالعملها. پیش‌بینی محتوای کَش دستورالعمل ممکن می‌باشد. اما پیش‌بینی کَش داده‌ای به سادگی نمی‌باشد. یک از مشکلات کَش فضای در دسترس محدود آن می‌باشد. همیشه بین نرخ برخورد و سرعت دستیابی به کَش، مصالحه[3] وجود دارد. کَش می تواند Direct Mapped یا Fully associative باشد [۵] [۸].

توازی سطح دستورالعمل[4]: بسیاری از پردازنده‌ها قادرند چند دستورالعمل را بصورت موازی اجرا نمایند. این موضوع نیز مشکلاتی برای تخمین زمان اجرای برنامه ایجاد می‌نماید. مشکل، نیاز به تعیین وابستگی‌های بین دستورالعملها قبل از زمان اجرا است تا معلوم گردد که چه دستوراتی می‌توانند بطور موازی اجرا گردند. این وابستگی می‌تواند وابستگی داده‌ای یا وابستگی ساختاری باشد. وابستگی ساختاری زمانیست که دو دستورالعمل برای تصاحب واحد اجرایی یکسان رقابت می‌نمایند و وابستگی داده‌ای زمانیست که دو دستورالعمل از ثبات یا حافظه مشترک استفاده می‌نمایند [۱۲].

---

[1] infeasible path

[2] Speculative Execution

[3] Trade off

[4] Instruction Level Parallelism(ILP)

آنچه که ضرروی می‌باشد اینست که بتوان وابستگی بین دستورات که سبب کاهش زمان اجرای آنها نسبت به حالتی که مستقل اجرا می-گردند را در ساختار پیشنهادی نشان داد.

۴-۳- دلایل استفاده از XML

XML قالبی فراگیر و جهانی است که برای نمایش اطلاعات ساخت یافته بصورت گسترده مورد استفاده قرار می‌گیرد. سند XML ذاتاً دارای ساختار سلسله مراتبی می‌باشد که برای نمایش ساختار کد برنامه مناسب می‌باشد. نمایش مبتنی بر XML بسادگی قابل فهم می‌باشد, بسادگی با استفاده از ابزارها قابل دستکاری بوده, بسیار منعطف, توسعه‌پذیر و بصورت گسترده پشتیبانی می‌گردد. مزایای نمایش مبتنی بر XML برای کد برنامه جاوا [۱۰] عبارتست از:

- ساختار کد صریح[۱] : سند XML ذاتاً ساخت‌یافته می‌باشد و می-تواند برای ارائه نمایشی از کد بشکل درختی مورد استفاده قرار گیرد.
- توانایی پرس‌وجو[۲] قدرتمند
- نمایش توسعه‌پذیر[۳]: کد برنامه بصورت متن- مسطح بسادگی قابل توسعه نیست. افزودن داده و کد جدیدی در کد متن- مسطح, سبب از بین رفتن ساختار کد شده و نیاز به تغییرات در تجزیه-کننده[۴] خواهد داشت. در حالی که در XML بدلیل توسعه‌پذیری براحتی می‌توان اطلاعات جدید مورد نیاز را در آن درج کرد.
- قالب‌بندی منعطف
- ارجاع متقابل[۵]
- پشتیبانی گسترده

با توجه به آشنایی و بکارگیری بسیار XML از بیان شرح ساختار آن پرهیز نموده و تجربه‌های بهره‌گیری از این ساختار در بخش ۶ (کارهای مرتبط) پیگیری خواهد شد.

۵- تعریف اجزای ساختار پیشنهای

۱. **دستورات:** برای هر دستور یک زمانی مشخص خواهد شد که به عنوان صفت[۶] آن دستور بیان می‌گردد.

<StatementK Line=n Time=n > Statement </ StatementK>

۲. **دستورات ورودی:** در این نوع دستورات مقدار متغیری از ورودی خوانده می‌شود. برای این دستورات در محاسبات مقدارشان مورد نظر می‌باشد. بهمین منظور حداقل و حداکثر مقدار متغیر به عنوان یک عنصر[۷] اضافه می‌گردد.

<INStatementK Line=n Time=n > Statement
<VarNameK Min=n Max=n> VariableName </VarNameK>
…
</ StatementK>

۳. **دستورات فراخوانی متد:** در دستورات فراخوانی زمان اجرای متد بر حسب پارامترهای ارسالی به متد ممکن است متفاوت باشد. برای دستورات فراخوانی باید به جای استفاده از زمان اجرای مطلق از حداقل و حداکثر زمان اجرا استفاده نمود.

<CallStatementK Line=n MinTime=n MaxTime> Statement </CallStatementK>

۴. **بلاک:** به مجموعه ای از دستوراالعملها اطلاق می گردد که زمان اجرای آنها در محاسبات نهایی, بصورت مجموع زمان اجرای آنها بطور مستقل لحاظ نمی‌گردد و زمان مشخص شده برای بلاک در نظر گرفته می شود. یک بلاک به عنوان یک عنصر[۸] درنظر گرفته شده و اطلاعات زمانی بلاک بعنوان صفات آن مطرح می‌گردد. دستورات داخل بلاک به عنوان فرزندان عنصر بلاک خواهند بود.

<BlockN …>
<StatementK …. > …. </StatementK>
…..
</BlockN>

بر اساس تعریف فوق بلاکهای مختلفی ایجاد خواهد شد :

بلاک ساده : در یک بلاک ساده دستوراتی با زمان مشخص وجود دارد. زمان اجرای بلاک ساده بر اساس مجموع زمان اجرای دستورات داخل آن محاسبه می گردد. برچسب آغازین آن بصورت زیر خواهد بود:

<SimpleBlockN TotalTime=n>
<StatementK …. > …. </StatementK>
</SimpleBlockN >

بلاک حلقه تکرار : در یک حلقه تکرار و بر اساس دستورات داخل آن, زمان هر تکرار ممکن است متفاوت باشد. به همین منظور برای یک بلاک حلقه تکرار اطلاعات زمانی آن بصورت صفت برای عنصر بلاک بوده و شامل موارد زیر خواهد بود:

- کمترین زمان اجرای یک تکرار
- بیشترین زمان اجرای یک تکرار
- حداقل تعداد تکرار
- حداکثر تعداد تکرار
- کل زمان اجرای حلقه تکرار

<loopBlockK MinExeTimePItr=n MaxExeTimePIt=n MinItr=n MaxItr=n TotalTime=n >
….
</LoopBlockK>

بلاک دستورات شرطی: هر بلاک دستور شرطی از دو بلاک مربوط به بدنه then و بدنه else تشکیل خواهد شد. هر یک از بلاکهای then و else زمان اجرای آن بصورت صفت بیان می‌گردد و کل زمان دستور شرطی بصورت حداقل, حداکثر و کل بعنوان صفت عنصر بلاک if خواهد بود.

<IfBlockK MinTime=n MaxTime=n TotalTime=n>

---

[1] Explicit Code Structure
[2] querying
[3] Extensible Representation
[4] Parser
[5] Cross-referencing
[6] attribute
[7] element
[8] element

```
    <ThenBlockK Time = n >
        .......
    </ThenBlockK>
    <ElseBlock Time=n >
        .......
    </ElseBlockK>
</IfBlockK>
```

بلاک دستورات انتخابی: در دستورات انتخابی(switch) هر case زمان مربوط خود را خواهد داشت. زمان هر case بعنوان صفت عنصر بلاک case بیان خواهد شد و کل زمان دستور انتخابی بصورت حداقل, حداکثر و کل بعنوان صفت عنصر بلاک switch بیان می‌گردد.

```
<SwitchBlockK  MinTime=n MaxTime=n  TotalTime=n>
        <CaseBlockK Time=n>
        ....
        </CaseBlockK>
        ....
</SwitchBlockK>
```

بلاک دستوراتی که عوامل سطح پایین بر آن موثرند: همانطور که در بخش فوق بیان شد عواملی مانند امکانات پردازنده‌های مدرن سبب شده که چند دستور همزمان اجرا شده یا دستورات سریعتر از زمان اجرای مطلق آن اجرا گردد. این دستورات در مجاورت همدیگر قرار دارند. بنابراین با شناسایی این دستورات می‌توان آنها را در یک بلاک(HLETA[1]) قرار دارد. زمان این بلاک(قطعه‌ای از کد) بجای مجموع زمان دستورات محاسبه می گردد.

```
<HLETABlockK Time = n >
        <StatementK …>
        ....
</HLETABlockK>
```

۵. پارامترهای ورودی و متدها: در ابتدای هر متد باید اطلاعات زمانی آن مشخص باشد. به همین منظور باید بلاکی این اطلاعات را داشته باشد.

- امکان درج پارامترها و مقدارشان به صورت حداقل و حداکثر و تعداد پارامترها
- زمان اجرای کل متد

```
<MethodTimeInfoBlockK  TotalTime=n>
    <ParameterBlockK  Parameters=n>
        <ParamK Min= n Max=n > ParameterName </ParamK>
                ....
                ....
    </ParameterBlockK>
</MethodTimeInfoBlockK>
```

البته در متدهای بازگشتی[2] زمان اجرای متد بر اساس زمان اجرای یکبار آن و مقدار عمق فراخوانی محاسبه خواهد شد. بنابراین نیاز خواهیم داشت که بلاک اطلاعاتی متدهای بازگشتی متفاوت از بلاک متدهای دیگر باشد.

```
<RecMethodTimeInfoBlockK    TimePCall=n   CallDepth=n
TotalTime=n >
    <ParameterBlockK  Parameters=n>
        <ParamK Min= n Max=n > ParameterName </ParamK>
                ....
                ....
    </ParameterBlockK>
</RecMethodTimeInfoBlockK>
```

۶. مسیرهای اجرا نشدنی

برای هر مسیر که یک مسیر اجرا نشدنی شناسایی گردد, به تمام بلاکها یا عناصر روی این مسیر صفتی افزوده می‌شود که نشان دهنده اجرا نشدن آن باشد.

```
<BlockN Infeasible Path=True>
        …..
</BlockN>
```

۶- کارهای مرتبط

همانطور که دربخش ۱ (مقدمه) بیان گردید, در بسیاری از روشهای تحلیل WCET ابتدا اطلاعات لازم را استخراج نموده و سپس تحلیل و محاسبات خود را انجام می‌دهند. استفاده از نمایشی از کد که بتواند اطلاعات زمانی را در خود نگه دارد, برای اینگونه روشها مناسب می‌باشد. می‌توان بطور نمونه به [۱۳] و [۱۴] اشاره نمود.

ابزارها و روشهای مختلف از نمایش میانی برای دو منظور استفاده نموده‌اند: اول اینکه این نمایش میانی قابلیت و تسهیلاتی برای استخراج اطلاعات زمانی را فراهم آورد و امکان درج توضیحات دستی در آن فراهم شده باشد. دیگر اینکه بتوان خروجی ابزاری که این نمایش میانی را تولید کرده است, بعنوان ورودی ابزاری دیگر استفاده گردد.

یکی از این نمایشهای میانی NIC[3] می‌باشد. کد برنامه به زبان C را به قالب NIC تبدیل می‌گردد[۱۶]. NIC بعنوان نمایش میانی در چارچوب تحلیل WCET بکار گرفته شده است[۱۵]. تسهیل استخراج اطلاعات, ارتباط بین ابزارهای مختلف و نمایش تصویری نتایج از اهم کاربردهای این نمایش میانی بوده است.

نمایش میانی دیگر TCD-Code[4] می‌باشد که ساختاری مشابه XML داشته و در پژوهش‌هایی از آن بهره گرفته شده است. TCD گرچه خصائصی برای تحلیل WCET دارد, اما برای به اشتراک‌گذاری داده‌ها بین ابزارهای بکار رفته نیز استفاده شده است.

در [۷] از XST که قالب و ساختاری مشابه با XML دارد, برای نمایش درخت نحوی, استخراج اطلاعات و تحلیل WCET استفاده گردیده است و علاوه بر آن با استفاده از مرورگری نتایج نمایش داده می‌شود و قابل دستکاری نیز می‌باشد.

XAC [5] هم یک نمایش میانی است که برای نگهداری اطلاعات در ابزارهای تحلیل برنامه‌ها استفاده گردیده است[۱۷]. XAC دو هدف مهم دارد: قابلیت حمل (مستقل از بستر) و دیگر قابلیت توسعه‌پذیری برای نگهداری اطلاعات اضافی مورد نیاز ابزارهای تحلیلی. در [۱۸] از MAD استفاده شده که بر اساس XML می باشد. MAD برای بیان

---

[1] Hardware-Level Execution Time Analysis
[2] Recursion
[3] New Intermediate code
[4] Textual Code Description
[5] eXtensible Annotation Class

ویژگیهای پردازنده‌های مدرن, معنای دستورات و ... در این تحقیق مورد استفاده قرار گرفته است[۱۹].

XTC [1] که بر اساس شمای XML طراحی شده است به عنوان اشتراک گذارنده داده بین دو ابزار aiT و SymTA/S بکار گرفته شده است[۲۰]. aiT خروجی مبتنی بر XML که قابل مشاهده باشد را تولید می‌نماید.

۷- بحث و بررسی

ساختار نمایش پیشنهادی را در دو بخش مورد بررسی و ارزیابی قرار می‌دهیم. ابتدا در خصوص اینکه تا چه حد توانایی نگهداری اطلاعات زمانی و محاسبه زمان اجرا را دارا می‌باشد. از سویی دیگر ساختار نمایش پیشنهادی زمینه استفاده در تحلیل نحوی و معنایی برای تحلیل WCET را تا چه میزان دارا می‌باشد.

اولین بحث در تحلیل WCET تحلیل کد برنامه در سطح کد منبع[2](سطح بالا), کد object و یا کد ماشین می‌باشد. نمایش مبتنی بر XML برای تمامی سطوح نمایش کد برنامه ممکن می‌باشد. همچنین با توجه به ساختار XML می‌توان برای نمایش نگاشت بین سطوح مختلف نمایش کد از آن استفاده نمود.

استخراج اطلاعات از کد برنامه بصورت خودکار یا با استفاده از توضیحات دستی می‌باشد. همانطور که در [۲۱] آمده است با وجود پیشرفتهای فراوان در خودکارسازی استخراج اطلاعات, استفاده از توضیحات دستی اجتناب ناپذیر می‌باشد. اضافه کردن توضیحات دستی به کد متن- مسطح برنامه سبب خواهد شد تغییراتی در تجزیه‌کننده برای تحلیل نحوی صورت پذیرد[۱۰]. علاوه براین بر اساس آنچه که در [۶] آمده است, زبان توضیحات باید مستقل از ابزار و متدلوژی استفاده شده باشد و در تمام سطوح نمایش کد برنامه قابل اضافه کردن باشد. این نیازها براحتی توسط XML پاسخ داده می‌شود.

تبدیل کد برنامه‌های زبانهای برنامه‌نویسی مختلف به XML مسئله‌ایست که باید مورد توجه قرار گیرد. بیشتر متدولوژیها و ابزارها که روی کد برنامه به زبان سطح بالا عمل می‌نمایند, زمان اجرای برنامه‌ها به زبان Java و C++ را محاسبه می‌نمایند. همانطور که در ادامه آمده است, تبدیل برنامه‌های java و C++ به XML تجربه گردیده و امکان پذیر است.

همانطور که در بخش ۳-۱ بیان گردید باید ساختاری برای نمایش اطلاعات استخراج شده لحاظ گردد. با توجه به ساختار پیشنهادی که در بخش ۵ بیان گردید, نمایش مبتنی بر XML ساختاری مناسب برای ساختارهای مطرح شده در ۴-۱ و ۴-۲ خواهد بود.

با توجه به ساختار پیشنهای امکان محاسبات(بخش ۳-۳) برای دو روش مبتنی بر ساختار درختی و مبتنی بر مسیر اجرایی وجود دارد. اما برای محاسبه با روش IPET نیاز است که تغییراتی در ساختار پیشنهادی صورت گیرد.

همانطور که بیان شد یکی دیگر از اهداف این مقاله فراهم آوردن زمینه‌ای برای تحلیل کد و استخراج اطلاعات آن بر اساس نمایش میانی مبتنی بر XML می‌باشد. در ادامه به تحقیقاتی که برای تحلیل کد و استخراج ساختارهای نحوی و معنایی کد برنامه, از نمایشی مبتنی بر XML استفاده کرده‌اند, می‌پردازیم. تمرکز بر کاربردهایی است که بتوان آنرا برای تحلیل WCET توسعه داد.

JavaML 2 یکی از نمایشهای میانی کد برنامه‌های جاوا بر حسب درخت نحوی آن می‌باشد که برای تحلیل برنامه‌ها و شناسایی ساختارهای استفاده شده در آن مورد استفاده قرار می‌گیرد[۲۲]. JavaML2 نمایشی مبتنی بر XML از کد برنامه جاوا می‌باشد. در [۲۳] از JavaML برای مهندسی معکوس از کد جاوا به UML مورد استفاده شده است.

یک دیگر از نمایشهای مبتنی بر XML, srcML نام دارد که نمایشی از کد برنامه‌های C++ می‌باشد[۲۴]. هدف از ارائه آن استخراج ایستای اطلاعات از کد برنامه‌های C++ می‌باشد.

ابزار XMLizer امکان تبدیل کد برنامه به زبان برنامه‌نویسی پاسکال و جاوا را بر اساس درخت نحوی آنها به XML فراهم می‌آورد[۲۵].

در بسیاری از ابزارها و روشهای تحلیل WCET از گراف استفاده می‌گردد[۲۶], امکان تبدیل گراف به XML نیز وجود دارد. GXL نمایش مبتنی بر XML را برای توصیف گراف فراهم می‌کند[۲۷].

بر اساس مطالب فوق می‌توان ساختار پیشنهادی را به نحوی توسعه داد که بتوان از آن برای تحلیل نحوی و شناسایی ساختار برنامه استفاده نمود.

نمونه اولیه‌ای از ساختار پیشنهادی در پژوهشهای [۱] [۲] پیاده‌سازی و مورد استفاده قرار گرفت.

۸- نتیجه‌گیری و کارهای آتی

محاسبه بیشترین زمان اجرای برنامه‌ها, گام مهم و ضروری در فرایند توسعه و تایید صحت سیستم های بی درنگ سخت می باشد. در روش تحلیل ایستا برای تخمین بیشترین زمان اجرای برنامه، باید طولانی‌ترین مسیر اجرایی برنامه مشخص گردد و بر اساس زمان اجرای هر دستورالعمل که بر روی این مسیر قرار دارد، می‌توان بیشترین زمان اجرای برنامه را تخمین زد. زمان اجرای هر دستورالعمل به کامپایلر و سخت‌افزاری که آنرا اجرا می‌نماید، بستگی دارد. برای مشخص نمودن طولانی‌ترین مسیر اجرای برنامه باید پیش‌بینی انشعابها، حدود تکرار حلقه‌ها و عمق فراخوانیهای بازگشتی را بررسی و مشخص نماییم. برای داشتن ابزاری که تخمین مناسبی از بیشترین زمان اجرای برنامه ارائه دهد, نیاز به در نظر گرفتن و پیاده سازی همه مسائل مطروحه می‌باشد.

---

[1] XML Timing Cookies
[2] source code

عدم امکان پیاده‌سازی همه بخشها مانع پیشرفت پژوهش در این حیطه می‌باشد. ارائه نمایشی از کد برنامه در قالب XML راه حلی برای مشکل فوق می باشد. ابزارهای تحلیل زمانی می توانند روی این نمایش عمل نموده و اطلاعات زمانی کد برنامه را که بدست آمده است در آن درج نمایند. اگر بخشی از کار توسط روش و ابزاری دیگر انجام گرفت و یا استخراج اطلاعات با یک روش خاص بر روی بستر سخت‌افزاری دیگری صورت گرفت[1]، این نمایش امکان بروزرسانی اطلاعات بدست آمده از روش مورد نظر را فراهم می‌آورد. استقلال از ابزار و بستر اجرایی از خصوصیات این نمایش است و می‌توان در برنامه‌های محک[2] برای ارزیابی و مقایسه روشها از آن استفاده گردد. همچنین امکان افزودن توضیحات دستی فراهم گردیده است.

در پژوهشهای بعدی این نمایش به نحوی توسعه خواهیم داد که بتوان تحلیل را مستقیم بروی نمایش ارائه شده انجام دهد. از سویی امکاناتی به آن افزوده شود تا بتوان اطلاعات معماری سخت‌افزار مورد استفاده در آن درج گردد و ضعف موجود در نگهداری اطلاعات زمانی در بلاک مربوطه به ساختارهای تاثیر پذیرفته از عوامل سطح پایین مرتفع گردد.

۹- سپاسگزاری

این پژوهش با حمایت مالی حوزه معاونت پژوهشی دانشگاه ملایر صورت پذیرفته است که جای تشکر و قدردانی دارد.

۱۰- مراجع

۱- سخائی‌نیا مهدی، "طرح و پیاده سازی محیطی برای تخمین خودکار طولانی ترین زمان اجرای برنامه جهت ارزیابی میزان تسریع حاصل از توزیع کد"، پایان-نامه برای دریافت درجه کارشناسی ارشد، دانشگاه علم وصنعت ایران، تهران، ایران، ۱۳۸۵

۲- پارسا سعید، سخائی‌نیا مهدی، " تخمین ایستای تعداد تکرار حلقه‌ها جهت پیش بینی زمان اجرای برنامه‌ها"، سیزدهمین کنفرانس ملی کامپیوتر انجمن کامپیوتر ایران، کیش، ایران، ۱۳۸۶

3- A. Ermedahl, "A Modular Tool Architecture for Worst-Case Execution Time Analysis", PhD dissertation, Dept. of Information Technology, Uppsala Univ. Uppsala, Sweden, 2003.

4- Suhendra, V. Mitra, T. Roychoudhury, A. T. Chen, "Efficient Detection and Exploitation of Infeasible Paths for Software Timing Analysis", Design Automation Conference, 2006 43rd ACM/IEEE, 2006.

5- J. Gustafsson, "Analysing Execution-Time of Object-Oriented Programs using Abstract Interpretation", PhD thesis, Uppsala University, Uppsala, Sweden, May 2000.

6- Raimund Kirner, Jens Knoop, Adrian Prantl, Markus Schordan, and Ingomar Wenzel, "TOWARDS A COMMON WCET ANNOTATION LANGUAGE: ESSENTIAL INGREDIENTS", In Proc. 8th International Workshop on Worst-Case Execution Time Analysis, Prague, July 2008.

7- G. Bernat, A. Colin, S. M. Petters, "pWCET: a Tool for Probabilistic Worst-Case Execution Time Analysis of Real-Time Systems", Technical Report YCS-2003-353 , Department of Computer Science, University of York, UK, February 2003

8- T. Lundqvist, "A WCET Analysis Method for Pipelined Microprocessors with Cache Memories", PhD thesis, Dept. of Computer Engineering, Chalmers University , Sweden, 2002.

9- R. Kirner, "Extending Optimising Compilation to Support Worst-Case Execution Time Analysis", PhD Thesis, Institut für Technische Informatik, Wien, 2003.

10- Hrvoje Simic, Marko Topolnik, "Prospects of encoding Java source code in XML" In Proc. of the 7th International Conference on Telecommunications, Zagreb, Croatia, 2003.

11- C. A. Healy, M. Sjodin, D. B. Whalley, "Bounding Loop Iterations for Timing Analysis", In Proc. IEEE Real-Time Technology and Aplications Symposium, pages 12–21,1998.

12- P. Sandström, "A look at Execution Time Analysis and Measuring Interrupt Latency", Master Thesis, Mälardalen University, 2000.

13- C. Healy, M. Sjodin, V. Rustagi, D. Whalley, and R. van Engelen, "Supporting timing analysis by automatic bounding of loop iterations", Real-Time Systems, 18(2/3):121{148, May 2000.

14- C. Ferdinand, R. Heckmann, M. Langenbach, F. Martin, M. Schmidt, H. Theiling, S. Thesing, R. Wilhelm, "Reliable and precise WCET determination for a real-life processor". In Proceedings of the First International Workshop on Embedded Software, LNCS 2211, p469{485. Springer, 2001.

15- Jan Gustafsson, Andreas Ermedahl, and Bjorn Lisper, "Towards a flow analysis for embedded system C programs", In Proc. 10th IEEE International Workshop on Object-oriented Real-time Dependable Systems (WORDS 2005), February 2005

16- A Retargetable Compiler for ANSI C website. URL: http://www.cs.princeton.edu/software/lcc/

17- E. Y.-S. Hu, G. Bernat, and A. J. Wellings, "A Static Timing Analysis Environment Using Java Architecture for Safety Critical Real-Time Systems", Proc. of the 7th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems WORDS-2002, p 77–84, January 2002.

18- Kaiyu Chen , Sharad Malik , David I. August, "Retargetable static timing analysis for embedded software", Proceedings of the 14th international symposium on Systems synthesis, Canada, 2001.

19- K. Keutzer, S. Malik, A. R. Newton, J. Rabaey and A. Sangiovanni-Vincentelli. "System Level Design: Orthogonolization of Concerns and Platform-Based Design". IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 19(12), December 2000.

20- Daniel Kästner, Reinhard Wilhelm, Reinhold Heckmann, Marc Schlickling, Markus Pister, Marek Jersak, Kai Richter, Christian Ferdinand: Timing Validation of Automotive Software. ISoLA 2008: 93-107, 2008

21- Kirner, R., Knoop, J., Prantl, A., Schordan, M. and Wenzel, I. WCET Analysis: The Annotation Language Challenge. In Post-Workshop Proceedings of the 7th International Workshop on Worst-Case Execution Time Analysis, Pisa, Italy, July 3, 2007.

22- Ademar Aguiar, Gabriel David, and Greg Badros, "JavaML 2.0:Enriching the Markup Language for Java Source Code", XML: Aplicacoes e Tecnologias Associadas (XATA 2004), Porto, Portugal, 2004.

23- C.R. Russell, R.G. Dewar, "XML Encoded Reverse Engineering of Java to UML", Technical Report HW-MACS-TR-0007, 2003

24- Michael L. Collard, Huzefa H. Kagdi and Jonathan I. Maletic, "An XML-based Lightweight C++ Fact Extractor" International Workshop on Program Comprehension, 2003.

25- Gregory McArthur, John Mylopoulos and Siu Ng. An Extensible Tool for Source Code Representation Using XML. Working Conference on Reverse Engineering, 2002.

26- C. Sandberg , A. Ermedahl , J. Gustafsson , B. Lisper, "Faster WCET flow analysis by program slicing", ACM SIGPLAN Notices, v.41 n.7, July 2006

27- R. C. Holt, A. Winter and A. Schürr, "GXL: Towards a Standard Exchange Format", Working Conference on Reverse Engineering, 2000.

[1] retargetability
[2] benchmark

کارگاه‌های آموزشی مرکز اطلاعات علمی

آموزش مهارت های کاربردی در تدوین و چاپ مقاله

اصول تنظیم قراردادها

مقاله نویسی علوم انسانی