به روز رسانی سیستم‌های نرم‌افزاری و کاهش هزینه‌های ناشی از آن

سیدحبیب سیف‌زاده

فهرست مطالب

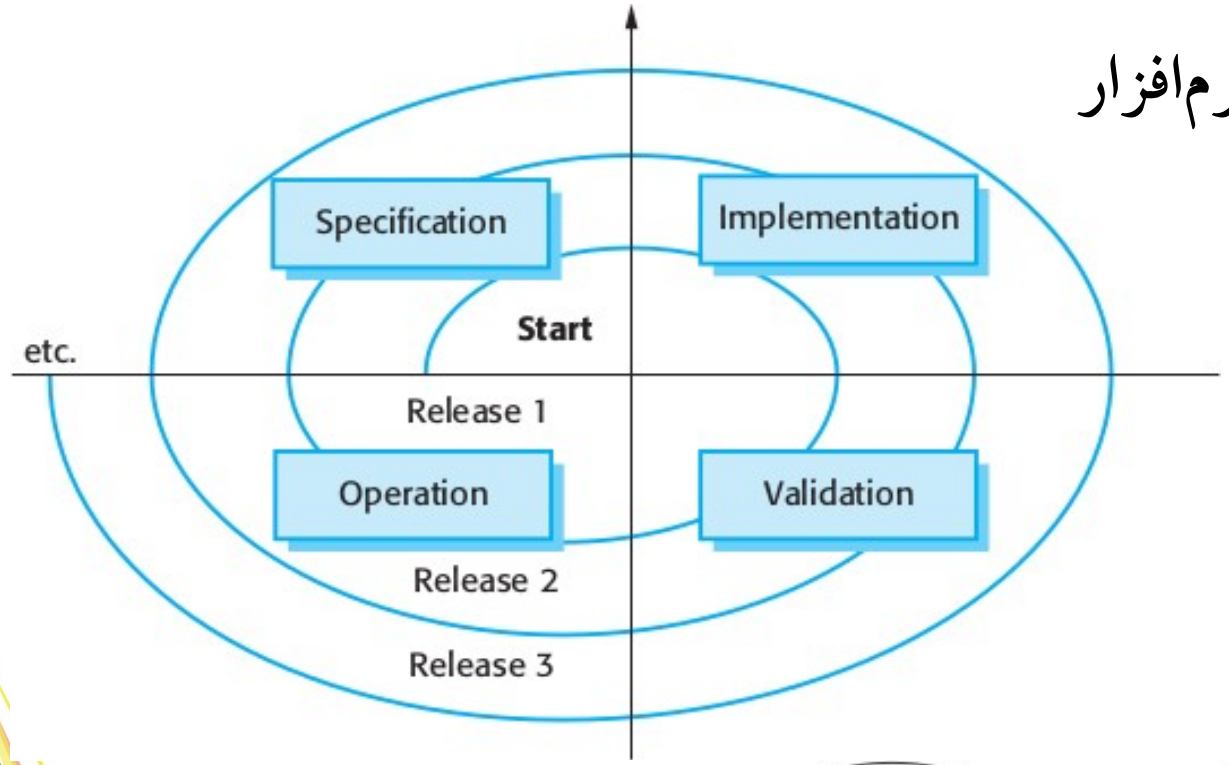
- مفاهیم اولیه
- اثبات به روز بودن موضوع
- زمینه‌های تحقیقاتی در حوزه تکامل نرم‌افزار
- نیازمندی‌ها
- توصیه‌های من

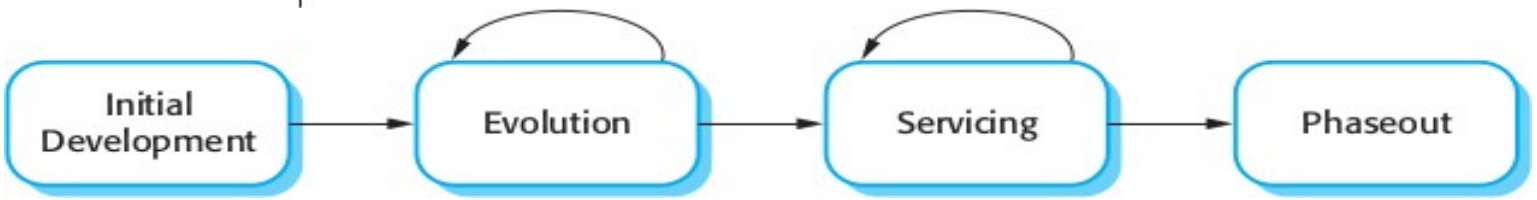
چرا تکامل؟

- طول عمر نرم‌افزارهای نظامی یا کنترل هوایی بیش از ۳۰ سال گزارش شده است
- طول عمر نرم‌افزارهای تجاری بیش از ۱۰ سال گزارش شده است
- در طول این مدت، نرم‌افزارها باید به روز شوند. به دلیل:
  - نیازهای تجاری جدید
  - گزارش خطاها در برنامه
  - تغییرات در برنامه‌های دیگر که در محیط برنامه قرار دارند
- بر این اساس، تکامل جز جداناپذیر هر فرآیند نرم‌افزار است

چرا تکامل؟ (ادامه)

- مثال‌هایی از فرآیندهای نرم‌افزار



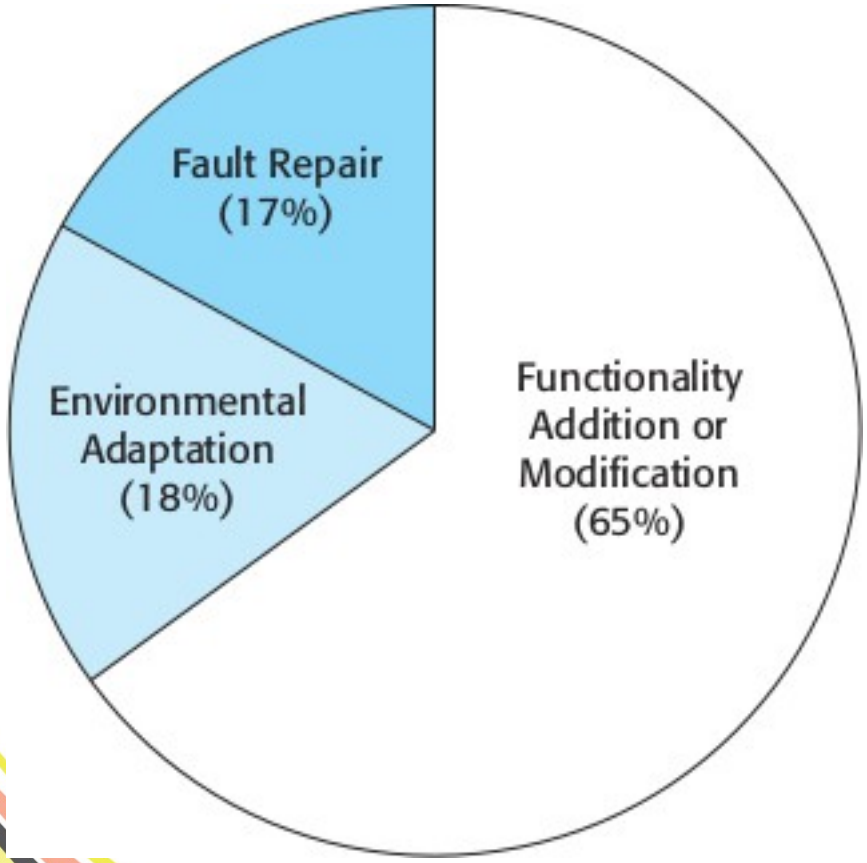


هزینه‌های ناشی از تکامل

- با توجه به طولانی بودن عمر نرم‌افزارها، سازمان‌ها نرم‌افزار را به عنوان یک سرمایه نگاه می‌کنند و برای تکامل آن هزینه‌های زیادی می‌پردازند
- قانون اول لمن: برای اینکه نرم‌افزار مفید باقی بماند، باید مرتبا تغییر پیدا کند
- بر اساس گزارش‌های مختلف، هزینه‌ای که در بخش تکامل نرم‌افزار انجام می‌شود بین ۸۵ تا ۹۰ درصد است
- نتیجه: کاهش هزینه تکامل نرم‌افزار، کاهش زیادی در کل هزینه یک نرم‌افزار ایجاد خواهد کرد

هزینه‌های ناشی از تکامل (ادامه)

- تغییراتی که در تکامل برنامه‌ها اتفاق می‌افتد معمولا شامل موارد زیر است:
  - اضافه نمودن ویژگی جدید
  - رفع خطا
  - منطبق کردن با محیط جدید



واژگان مورد استفاده

- تفاوت تکرار (Iteration) با تکامل (Evolution): هرگاه در اول تکرار، فهم نرم‌افزار وجود انجام شود تکامل رخ داده است
- نگهداری (Maintenance): به تکاملی اطلاق می‌شود که معمولا توسط گروهی غیر از توسعه‌دهنده اصلی نرم‌افزار انجام گیرد (فهم نرم‌افزار مهم تر است)
- بازسازی مجدد (Refactoring): تکنیک‌هایی که باعث می‌شوند ساختار برنامه‌ها بهبود یابند

واژگان مورد استفاده (ادامه)

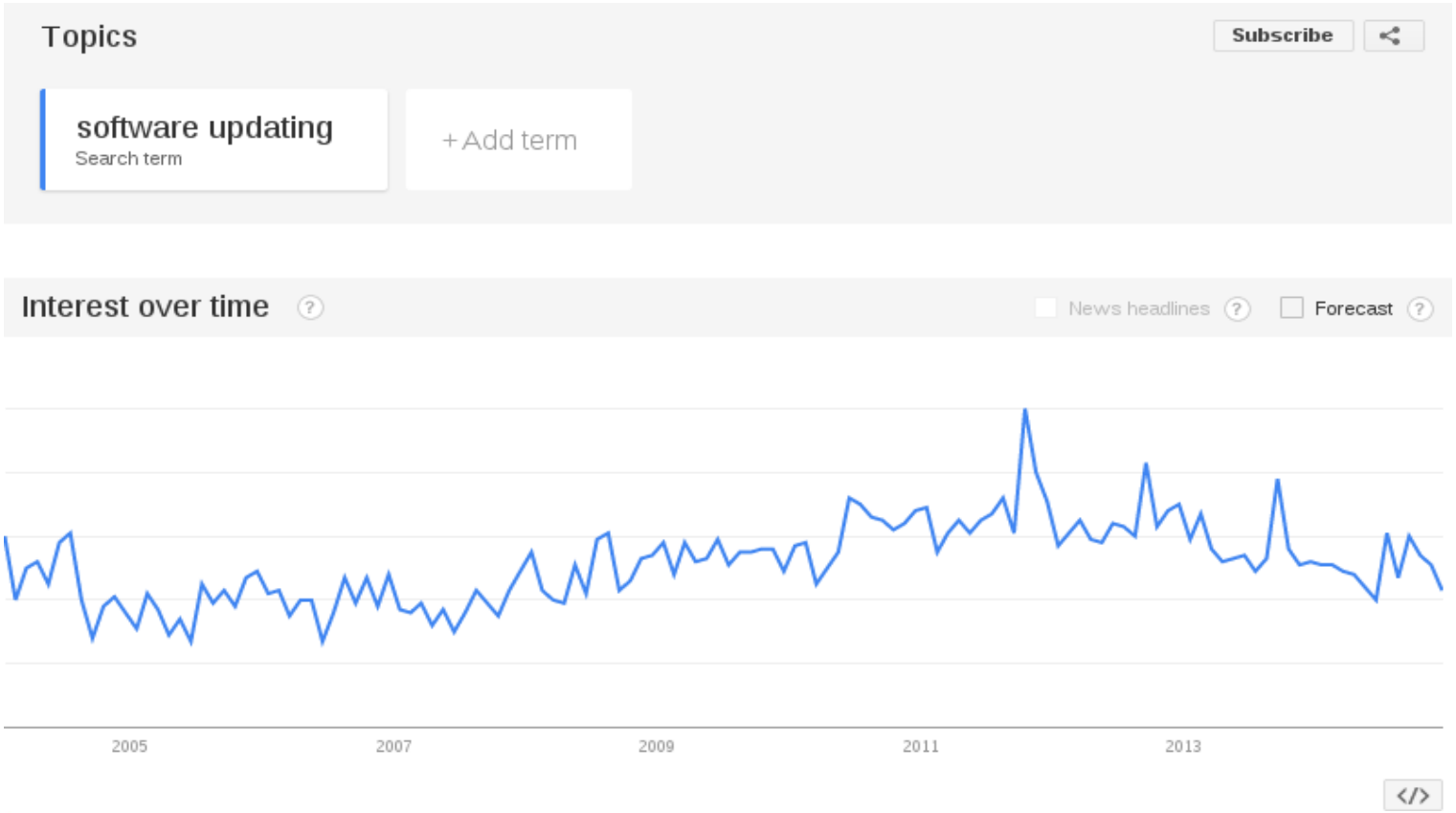
- مهندسی مجدد (Re-engineering): گسترده‌تر از بازسازی مجدد است، شامل به روز رسانی مستندات برنامه و مهندسی مجدد داده‌ها نیز می‌باشد
- نرم‌افزارهای موروثی (Legacy Systems): نرم‌افزارهایی که به دلیل به روز شدن محیط اطراف، امکان تغییر در آنها وجود ندارد ولی برای سازمان مفید هستند
- بدهی‌های فنی (Technical Debt): کدهای ناقص، تکراری یا دارای خطایی که در کد اعمال شده‌اند تا یک به روز رسانی حیاتی و سریع اتفاق افتد. این کدها بعدا باید بازسازی مجدد شوند

واژگان مورد استفاده (ادامه)

- جستجوی کد (Code Search): کاویدن کد به منظور یافتن ساختار و معنای برنامه
- کپی (Clone): کدهای تکراری در سراسر برنامه
- به روز رسانی (Updating): تکامل نرم‌افزار را از دید برنامه‌نویسی و کدنویسی مطرح می‌نماید

آیا تکامل یا نگهداری نرم‌افزار قدیمی شده است؟

- زمینه‌های تحقیقاتی مرتبط با مهندسی نرم‌افزار از ابتدای علم نرم‌افزار مورد علاقه محققان بوده و هنوز هم دنبال می‌شود



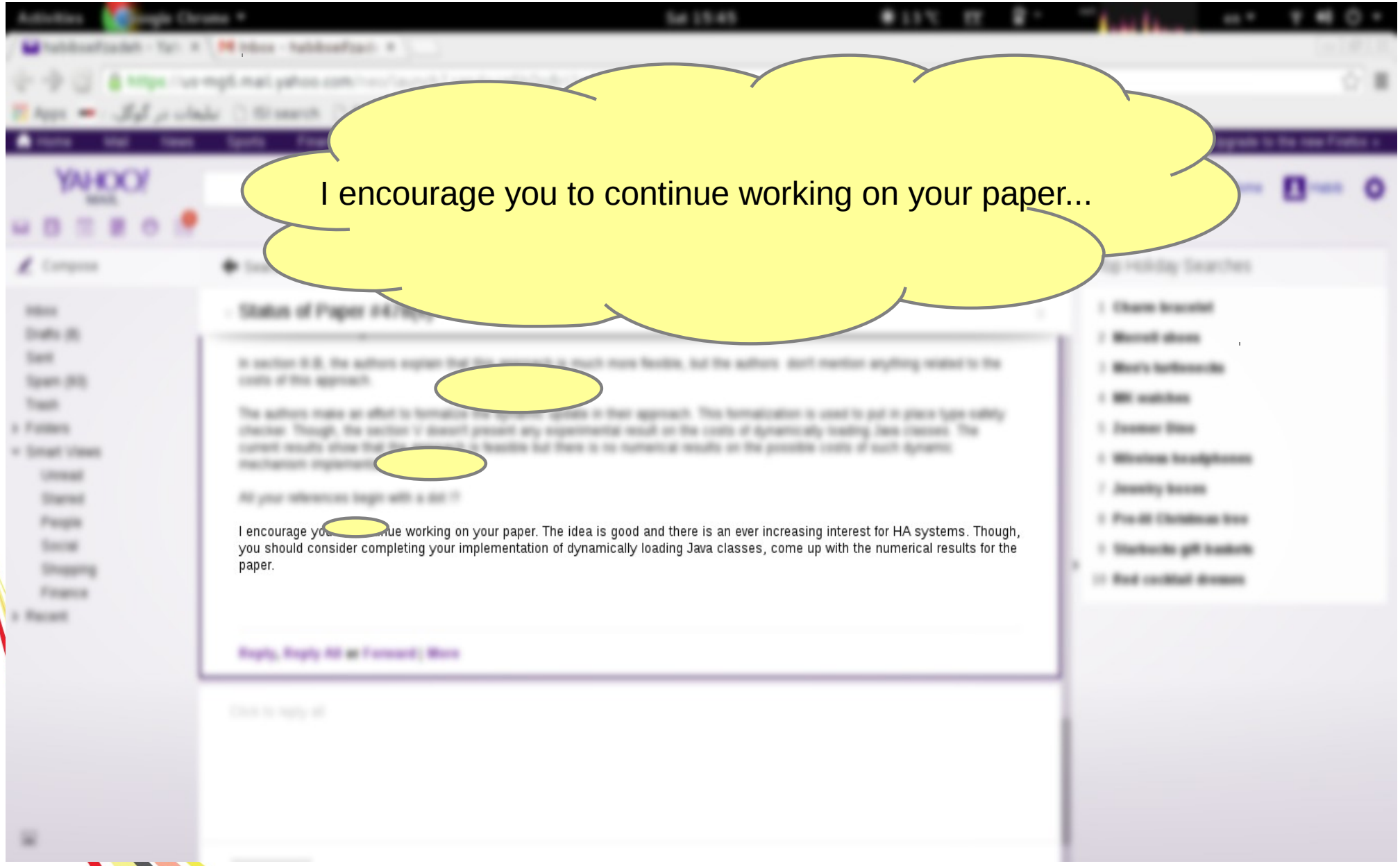
منابع در زمینه تکامل نرم‌افزار

- ICSME (International Conference of Software Maintenance and Evolution)
- EuroSPI (European Conference on Software Process Improvement)
- CSMR (European Conference on Software Maintenance and Re-engineering)
- WCRE (Working Conference on Reverse Engineering)

منابع در زمینه تکامل نرم‌افزار

- ICSE (International Conferenc of Software Engineering)
- ARES (Ava ecurity)
- HotS tware Upgrad
- Jou ocess

آیا با این همه منبع که مرتبا در حوزه نگهداری نرم‌افزار مقاله چاپ می‌کنند، این حوزه قدیمی است؟


I encourage you to continue working on your paper...
I encourage you to continue working on your paper. The idea is good and there is an ever increasing interest for HA systems. Though, you should consider completing your implementation of dynamically loading Java classes, come up with the numerical results for the paper.

زمینه‌های کاری

- ایجاد ابزار
  - بهبود کارآیی محیط‌های برنامه‌سازی
- شناسایی محل دقیق شکست
  - کاویدن رد پشته برای شناسایی محل شکست
  - کاویدن گزارش‌های خطا برای یافتن محل شکست
- تست
  - ایجاد ابزار تست خودکار
  - جستجوی کد به منظور تولید خودکار نمونه‌های تست

زمینه‌های کاری (ادامه)

- تخمین هزینه نگهداری
  - تخمین هزینه هر بخش از برنامه از طریق پیچیدگی آن
  - پیش‌بینی میزان تغییرات با استفاده از تاریخچه تغییرات گذشته
- به روز رسانی پویا
  - به روز رسانی برنامه‌های وب به صورت پویا
  - به روز رسانی پویای سیستم های عامل موبایل
- مهندسی مجدد و مهندسی معکوس
  - بهبود خودکار ساختار برنامه یا استخراج خودکار مستندات از کد
  - شناسایی الگوهای مناسب مهندسی مجدد از طریق کاویدن کدهای منبع

زمینه‌های کاری (ادامه)

- کپی
  - شناسایی هرچه بیشتر و دقیق‌تر کدهای تکراری در برنامه
  - شناسایی و معرفی کپی‌های معروف در برنامه‌ها
- مدیریت تغییر
  - کدام برنامه‌نویسان بیشتر با هم تعامل دارند؟
  - در نسخه بعدی نرم‌افزار، احتمال تغییر کدام قطعات بیشتر است؟
- مطالعات تجربی
  - شناسایی خودکار پست‌های کیفیت پایین یا تکراری در StackOverflow
  - شناسایی سئوالات متداول برنامه‌نویسان به وسیله کاویدن Stack...

افراد فعال در حوزه نگهداری نرم‌افزار

- Michael Hicksبه روز رسانی پویا، دانشگاه مری‌لند. سال ۲۰۰۱ در زمینه به روز رسانی پویا دکتری گرفته است
- Iulian Neamtiuبه روز رسانی پویا، دانشگاه کالیفرنیا. سال ۲۰۰۸ در زمینه به روز رسانی پویا دکتری گرفته است
- C Guiffrida سال ۲۰۱٤ در زمینه به روز رسانی پویا دکتری گرفته است. استاد راهنما تننباوم
- Gregerson سال ۲۰۱۲ در زمینه به روز رسانی پویا دکتری گرفته است

افراد فعال در حوزه نگهداری نرم‌افزار (ادامه)

- Tom Mens استاد تمام دانشگاه UMONSبلژیک . زمینه کاری: تکامل نرم‌افزار
- Grardo Canfora استاد تمام دانشگاه Sannioایتالیا. زمینه کاری: مهندسی نرم‌افزار. سردبیر مجلهJournal of Software: Evolution and Process
- Michael Collardکاویدن کدهای منبع

نیازمندیها

- آشنایی با مفاهیم مهندسی نرم‌افزار خصوصا تست و نگهداری (فصول مربوطه از کتاب‌های پرسمن و سامرویل مطالعه شود)
- آشنایی با نرم‌افزارهای کنترل نسخه مانند Git، SVN و ...
- آشنایی با نرم‌افزارهای تحلیل کد منبع مانند Lex و Yacc
- توانایی خواندن مقالات انگلیسی

نیازمندیها (ادامه)

- تسلط به برنامه‌نویسی (هر زبانی که قرار است ایده در آن پیاده‌سازی یا ارزیابی شود)
  - آشنایی با زبان C (محیط gcc) و جاوا بیش از بقیه توصیه می‌گردد
- آشنایی با زبان‌های برنامه‌سازی مختلف مانند Erlang
- آشنایی با خطایاب‌ها مانند gdb و jdb
- آشنایی با نرم‌افزارهای ردیابی خطا مانند BugZilla

توصیه‌های من

- اولین و مهم‌ترین قدم برای انجام هر تحقیق: مطالعه، مطالعه، مطالعه
- برای این کار از همایش‌ها و مجلاتی که در این سخنرانی معرفی شد بهره بگیرید
- حتی‌الامکان از منابع فارسی استفاده نکنید
- مقالات را به صورت کامل ترجمه نکنید، بلکه سعی کنید ایده اصلی آن را استخراج کنید (خواندن هر مقاله نباید بیش از دو سه روز طول بکشد)
- اگر خواندن مقالات وقت زیادی از شما می‌گیرد به دنبال علت و سپس مرتفع کردن آن باشید (به عنوان مثال، تقویت زبان انگلیسی)
- خواندن مقاله‌ای را که نمی‌فهمید به زمان دیگری موکول کنید

توصیه‌های من (ادامه)

- پس از خواندن هر مقاله، شخصی که مقاله را نوشته، منابعی که استفاده کرده و کارهای دیگری که به آن مقاله ارجاع داده‌اند را جستجو نمایید
- هنگام خواندن هر مقاله، دید انتقادی به آن داشته باشید
- به نحوه ارزیابی هر مقاله دقت کنید. ارزیابی نباید بر روی یک سازمان کوچک یا نرم‌افزاری که کسی آن را نمی‌شناسد انجام گیرد
- ارزیابی‌ها در حوزه نگهداری نرم‌افزار معمولا بر روی نسخه‌های متعددی از نرم‌افزارهای متن باز معروف مانند آپاچی، فایرفاکس و یا نرم‌افزارهایی که در sourceforge پیدا می‌شوند انجام می‌گیرد

توصیه‌های من (ادامه)

- به بخش کارهای آینده هر مقاله اهمیت خاص بدهید هرچند که نباید تنها به آنها اکتفا نکنید
- از کلیه مقالاتی که خوانده‌اید گزارشی تهیه نمایید
- تا چنین گزارشی را آماده نکرده‌اید سراغ ایده‌پردازی نروید
- به همه چیز شک داشته باشید، مثال: از خود بپرسید آیا اجتناب از coupling در برنامه‌های موبایل نیز باید انجام شود؟ آیا در برنامه‌های حال حاضر موبایل چنین است؟
- ایده شما ارزیابی چنین پرسشی خواهد بود
- به ایده‌های خود مطمئن باشید

توصیه‌های من (ادامه)

- این طور که مقالات حوزه نگهداری نرم‌افزار را بررسی کرده‌ام، کارهای تحقیقاتی در این حوزه با یک پرسش تحقیقی آغاز می‌گردند
- و با مطالعه تجربی بر روی حجم وسیعی از کدهای منبع ارزیابی می‌شوند
- تحلیل کد منبع و ابزارهایی که برای این منظور تولید شده‌اند اهمیت ویژه‌ای در حوزه نگهداری نرم‌افزار دارند
- تجربه کاری در حوزه مهندسی نرم‌افزار مهم است
  - شخصی که در فروش و نگهداری چند نرم‌افزار شرکت داشته، اهمیت تست، کاهش هزینه نگهداری یا استفاده از نرم‌افزار کنترل نسخه را بهتر از شخصی فاقد این تجربه درک می‌کند

سئوال؟

با تشکر