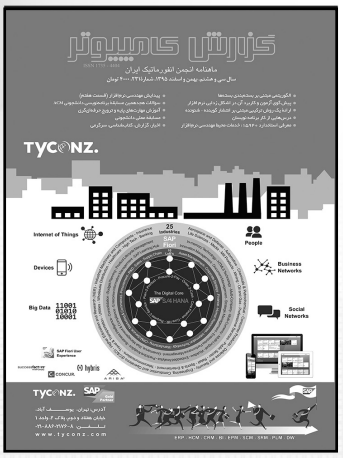
ماهنامهٔ انجمن انفورماتیک ایران
(فعلاً هردوماه یکبارمنتشرمی شود.)

گزارش کامپیوتر

سال سی و هشتم، بهمن و اسفند ۱۳۹۵ ، شمارهٔ ۲۳۱، ۴۰۰۰ تومان



صاحب امتیاز: انجمن انفورماتیک ایران
مدیر مسئول و سردبیر: ابراهیم نقیب‌زاده مشایخ

- مقاله
  - **الگوریتمی مبتنی بر بســـته‌بندی بسته‌ها برای مدیریت بهتر منابع در محیط ابر-** زهرا موحدی‌نیا، محمدرضا خیام باشی — ۸
  - **پیش‌گوی آزمون و کاربرد آن در اشکال‌زدایی نرم‌افزار-** علیرضا خلیلیان، مریم هاشم‌زاده، احمد براآنی دستجردی، بهمن زمانی — ۱۴
  - **ارائهٔ یک روش ترکیبی مبتنی بر انتشـــار گوینده شنونده و بلاندل برای تشخیص جوامع به صورت پویا در شبکه‌های اجتماعی-** محمد ستّاری، کامران زمانی‌فر — ۳۶
  - **درس‌هایی از کارِ برنامه نویسان-** علیرضا خلیلیان — ۵۰
- گزارش
  - **مسابقه عملی دانشجویی** — ۱۷
  - **سؤالات هجدهمین مسابقهٔ منطقه‌ای برنامه‌نویسی دانشجوئی ACM-** کیان اهرابیان — ۵۸
- بخش‌ها
  - **اخبار (انجمن، ایران، جهان)** — ۲
  - **خواندنی- پیدایش مهندسی نرم‌افزار (قسمت هفتم)-** ابراهیم نقیب‌زاده مشایخ — ۲۰
  - **دیدگاه- ترویج مدل بلوغ و پای‌بندی به استفاده از بهترین تجارب-** سیدابراهیم ابطحی — ۴۲
  - **استاندارد - معرفی استاندارد ۱۵۹۴۰: خدمات محیط مهندسی نرم‌افزار-** سید علی آذرکار — ۴۳
  - **کتاب شناسی- آموزش مهارت‌های پایه و ترویج حرفه‌ای‌گری-** سید ابراهیم ابطحی — ۵۵
  - **سرگرمی** — ۶۴

مقاله‌ها و گزارشهای تالیفی یا ترجمه خود را برای گزارش کامپیوتر بفرستید. مطالب ارسالی را با خط خوانا (یا ماشین شده) بر یک روی کاغذ بنویسید فاصله کافی برای افزودن اصلاحات ویرایشی در بین خط‌ها و حاشیه در نظر بگیرید. در صورت ارسال مطلب ترجمه شده، یک نسخه از اصل مطلب را نیز ارسال دارید. شکلها باید با روان‌نویس مشکی ترسیم و به‌صورت آماده چاپ ارسال شوند. مطالب ارسالی پس فرستاده نمی‌شوند.

مسؤولیت مطالب گزارش کامپیوتر با نویسندگان است و لزوماً نشان‌دهنده نظر انجمن انفورماتیک ایران نیست. ذکر نام شرکتها و محصولات در مطالب گزارش کامپیوتر برای ارائه اطلاع به خوانندگان است و نباید به معنی تائید انجمن از آنها تلقی شود.

سازمان آگهی‌ها: مهناز خاوندکار
تلفکس : ۸۸۳۲۸۴۱۷-۲۱
حروفچینی : تارتن سامانه
لیتوگرافی: نگارین پرتو ۷۷۵۳۰۳۰۷
چاپ علوی تهران : ۶۶۷۰۱۵۲۷
خ جمهوری ، ۳۰ تیر ، کوچه مصری پور ، پلاک ۲۵۶

تهران- خیابان انقلاب ، خیابان وصال شیرازی، کوچه شفیعی، پلاک ۵، واحد ۷

اخبار انجمن

اتمام چاپ اوّل کتاب بازنگری در کار

با اســتقبال وســیعی کــه از ســوی جامعهٔ انفورماتیک کشــور به عمل آمــد، چاپ اوّل کتاب بازنگری در کار (۱۰۰۰ نســخه) ظرف مدت چهار ماه به پایان رســید. چاپ اوّل این کتاب در مهر ماه سال جاری از سوی انجمن انفورماتیک ایران و با حمایت مالی شــرکت پویا منتشر شده بود.

چاپ دوم این کتاب تا پیش از پایان ســال، از ســوی انجمن منتشر خواهد شد.علاقه‌مندان می‌توانند از هم اکنون برای سفارش کتاب با دفتر انجمــن (۶۶۴۱۲۸۶۱ و ۶۶۴۱۲۹۷۶) تماس بگیرند.

تغییر نشانی دفتر انجمن

دفتر انجمن انفورماتیک ایــران از تاریخ اول دی ماه ۱۳۹۵ به نشانی زیر تغییر مکان یافته است:

تهران- خیابــان وصال شــیرازی- پائین تر از چهارراه طالقانی- کوچهٔ شــفیعی- پلاک ۵- واحد ۷ شــماره تلفن‌هــای دفتر انجمن ۶۶۴۱۲۸۶۱ و ۶۶۴۱۲۹۷۶ است.

راه‌اندازی کانال خبری انجمن در تلگرام

کانال خبــری انجمن انفورماتیــک ایران در تلگرام راه‌اندازی شد. از این پس علاقه‌مندان می‌توانند عــلاوه بر وبگاه انجمــن، از طریق نشــانی https://telegram.me/isi_news نیز از اخبار انجمن آگاهی پیدا کنند.

تفاهم نامهٔ همکاری

یک تفاهم‌نامهٔ همکاری بین انجمن انفورماتیک ایران و انجمن مدیریت اجرایی ایران به امضاء رســید. هدف از امضاء این تفاهم‌نامه، ارتقاء ســطح دانش مدیریتی اعضای هر دو انجمن و همکاری‌های مشــترک برای توسعهٔ علم و دانایی در چارچوب موضوع فعالیت‌های هر دو انجمن است.

به موجب این تفاهم‌نامه، تعهدات مشترک زیر برای هر دو انجمن در نظر گرفته شده است:

۱- عضویت افتخاری هــر انجمن در انجمن دیگر به‌عنوان عضو حقوقی

۲- عضویــت افتخاری اعضاء هیئت مدیره هر انجمن در انجمن دیگر به‌عنوان عضو حقیقی (در صورت تمایل آنان و در دوره عضویت آنان در هیئت مدیره)

۳- معرفی هر انجمــن در وبگاه انجمن دیگر و امــکان لینک از وبگاه هــر انجمن به وبگاه انجمن دیگر

۴- مشــارکت در انجــام طرح‌هــای علمی و پژوهشی مشترک که با توافق طرفین تعریف می‌گردد

۵- برگــزاری انواع ســمینارها و کارگاه‌های آموزشی مشترک

۶- برگزاری جلسات مشترک سالیانه فی‌مابین اعضا هیئت مدیره هر دو انجمن

۷- حضــور رایــگان اعضــاء هــر انجمن در سمینارها و کارگاه‌های آموزشی که به صورت رایگان توسط انجمن دیگر برگزار می‌گردد.

۸- ارائه ۳۰٪ تخفیف در پرداخت حق عضویت به کلیــه اعضای حقیقی و حقوقی هر انجمن کــه تمایل دارند در انجمن دیگر عضو گردند. (پس از دریافت معرفی‌نامه از انجمن دیگر)

۹- ارائه تا ۳۰٪ تخفیــف به اعضا هر انجمن (بــا توافق طرفیــن)، برای ثبت‌نــام در کلیه کارگاه‌ها، سمینارها، همایش‌ها و سایر مواردی که توسط انجمن دیگر برگزار می‌گردد.

۱۰- اطلاع‌رســانی اخبار مهــم هر انجمن به انجمن دیگر از طریق ارســال پیام، ایمیل، و معرفی در وبگاه، نشــریات، خبرنامه‌ها و دیگر امکانات اطلاع‌رسانی

برگزاری دوره‌های تخصصی برای سازمان‌ها

گروه‌هــای تخصصــی انجمــن انفورماتیک ایران آمادگــی برگزاری دوره‌های زیر را برای کارشناسان سازمان‌های مختلف در محل آن سازمان دارند.

۱- کارگاه نیــم روزه تدویــن توافقنامه‌های سطح خدمات

۲- کارگاه یــک روزه طراحــی و اســتقرار

پیشخوان خدمات
۳- کارگاه نیم روزه طراحی کاتالوگ خدمات
۴- کارگاه یک روزه ابزارهای مدیریت خدمات فناوری اطلاعات
۵- کارگاه نیم روزه برون‌سپاری خدمات و محصولات فناوری اطلاعات
۶- کارگاه نیم روزه همسوسازی IT با کسب وکار (IT Alignment) از طریق COBIT5
۷- کارگاه یک روزه راهبری فناوری اطلاعات (IT Governance) از طریق COBIT5
۸- کارگاه یک روزه راهبری فناوری اطلاعات (IT Governance)
۹- کارگاه نیم روزه مدیریت تدوام کسب و کار ISO2012:22301
۱۰- کارگاه نیم روزه راهبری فناوری اطلاعات و ITBSC
۱۱- کارگاه نیم روزه تدوین راهبرد خدمات فناوری اطلاعات
۱۲- کارگاه یک روزه مدیریت خدمات فناوری اطلاعات
۱۳- کارگاه سه روزه ITIL V3 Foundation 2011
۱۴-کارگاه یک روزه مدیریت مخاطرات فناوری اطلاعات
۱۵- کارگاه سه روزه دوره جامع COBIT5
۱۶- کارگاه نیم روزه توسعه نیازمندی‌ها مبتنی بر چارچوب CCMI-ACQ و استاندارد ISO/IEC29148:2011
۱۷- دورهٔ دو روزهٔ اسکرام مستر حرفه‌ای
ویژگی‌های این کارگاه‌ها عبارتند از :
• مشارکت گروهی، بحث و انجام سناریو، حل نمونه آزمون‌های امتحانی
• استفاده از مثال‌های مرتبط با حوزه فعالیت هر کسب وکار
• برگزاری دوره حداقل توسط دو مدرس دارای مدرک بین‌المللی
• برگزاری دوره در محل سازمان متقاضی
• ارائه مدرک شرکت در دوره از سوی انجمن انفورماتیک ایران
• ارائه مستندات آموزشی
علاقه‌مندان می‌توانند با دفتر انجمن انفورماتیک ایران (۳-۸۸۸۶۱۴۲۱) و یا نشانی پست الکترونیکی member@isi.org.ir تماس بگیرند.

اخبار ایران

بیست و دومین کنفرانس ملی سالانهٔ انجمن کامپیوتر ایران

بیست و دومین کنفرانس ملی سالانهٔ انجمن کامپیوتر ایران از ۲۱ تا ۲۳ اسفند ۹۵ در دانشگاه صنعتی شریف تهران برگزار خواهد شد. این کنفرانس با همکاری مرکز راه‌کارهای اطلاعاتی هوشمند و پژوهشکدهٔ فناوری اطلاعات و ارتباطات پیشرفتهٔ دانشگاه صنعتی شریف برگزار می‌شود.
کنفرانس امسال در ۱۱ شاخهٔ اصلی زیر مقالات را بررسی و انتخاب خواهد کرد:
۱- مهندسی نرم‌افزار
۲- سیستم‌های هوشمند و رایانش نرم
۳- امنیت داده و شبکه‌های کامپیوتری
۴- سیستم‌های موازی و توزیعی
۵- فناوری اطلاعات و ارتباطات
۶- داده‌های حجیم و رایانش ابری
۷- الگوریتم و نظریهٔ محاسبات
۸- معماری کامپیوتر و سیستم‌های دیجیتال
۹- پردازش سیگنال
۱۰- شبکه‌های اجتماعی
۱۱- شبکه‌های کامپیوتری
علاقه‌مندان برای دریافت اطلاعات بیشتر می‌توانند به وبگاه کنفرانس به نشانی csicc2017.ir مراجعه کنند.

کنفرانس سالانهٔ الگوی اسلامی ایرانی پیشرفت

مرکز الگوی اسلامی ایرانی پیشرفت، ششمین کنفرانس سالانهٔ خود را با موضوع «تعمیق و تکمیل الگوی پایهٔ پیشرفت» در تاریخ ۱۳ و ۱۴ اردیبهشت ۹۶ برگزار می‌کند.
محورهای اصلی این کنفرانس عبارتند از:
۱- کلیات الگوی پایهٔ پیشرفت
۲- تعمیق و تکمیل پیش‌نویس مبانی
۳- تعمیق و تکمیل پیش‌نویس آرمان‌ها و رسالت
۴- تعمیق و تکمیل پیش‌نویس افق
۵- تعمیق و تکمیل پیش‌نویس تدابیر
۶- مقایسهٔ الگوی پایهٔ پیشرفت با دیگر الگوها
علاقه‌مندان برای دریافت اطلاعات بیشتر می‌توانند به وبگاه کنفرانس به نشانی confer-ence.olgou.ir مراجعه کنند.

همایش شهر هوشمند و اینترنت اشیاء

همایش شهر هوشمند و اینترنت اشیاء در تاریخ ۲۱ فروردین ۱۳۹۶ در دانشگاه فردوسی مشهد برگزار می‌گردد.
محورهای همایش عبارتند از:
- زیرساخت‌های ارتباطی شهرهای هوشمند
- کاربردهای ICT و خدمات شهر هوشمند
- اینترنت اشیاء در شهرهای هوشمند
- استانداردها و نقشه راه توسعه شهرهای هوشمند
- بومی‌سازی خدمات شهرهای هوشمند
- مدیریت ترافیک و حمل و نقل هوشمند
- مدیریت انرژی و هوشمندسازی شبکه‌های برق
- سلامت و محیط زیست هوشمند
علاقه‌مندان برای دریافت اطلاعات بیشتر می‌توانند به وبگاه همایش به نشانی http:// sciot2017.um.ac.ir مراجعه کنند.

کنگره بین‌المللی سیستم‌های کلان‌داده و کلان‌مقیاس محاسباتی

کنگره بین‌المللی Top IIPC2017 از ۴ تا ۶ اردیبهشت ۱۳۹۶ با همکاری دانشگاه ایتالیایی کالابریا و مرکز تحقیقات فوق پیشرفتهٔ اَبَرایانهٔ ایتالیا و مراکز علمی و تحقیقاتی داخل کشور مانند معاونت علمی و فناوری ریاست جمهوری، مرکز تحقیقات مخابرات و وزارت علوم، تحقیقات و فناوری در حوزهٔ سیستم‌های کلان‌داده و کلان‌مقیاس محاسباتی برگزار می‌شود.

حضور افراد صاحب نام و تراز اوّل خارجی در ایـــن حوزه‌ها در این کنگره پیش‌بینی شـــده اســـت. این کنگره تنها رویداد مشـــترک بین اروپا و آسیاســـت که منحصـــراً بر موضوعات کلان‌داده‌هـــا و سیســـتم‌های محاســـباتی کلان‌مقیاس متمرکز است.
علاقه‌مندان برای کســـب اطلاعات بیشتر به وبگاه کنگـــره به نشـــانی www.tophpc.com مراجعه کنند.

دومین همایش ملی الگوریتم‌های فراابتـــکاری و کاربردهای آن در علوم و مهندسی

دانشـــگاه پیام نور نجف‌آباد، دومین همایش ملی الگوریتم‌های فراابتـــکاری و کاربردهای آن در علوم و مهندســـی را در اردیبهشت ماه ۱۳۹۶ برگزار می‌کند.
محورهای این همایش عبارتند از:
۱- تحلیل رفتاری و مبانی ریاضی الگوریتم‌های فرا ابتکاری
۲- الگوریتم‌های فـــرا ابتکاری الهام گرفته از طبیعت
۳- الگوریتم‌های اَبَر ابتکاری
۴- الگوریتم‌هـــای فـــرا ابتـــکاری در علوم و مهندسی
۵ - کاربـــرد الگوریتم‌هـــای فـــرا ابتکاری در بهینه‌سازی
۶- محاسبات تکاملی و هوش جمعی
۷- یادگیری ماشین و محاسبات نرم
۸- سایر موضوعات مرتبط
علاقه‌منـــدان برای کســـب اطلاعات بیشـــتر می‌توانند به وبگاه همایش به نشـــانی www.mha2017.ir مراجعه کنند.

کنفرانس فنـــاوری اطلاعات در مدیریت شهری

شهرداری تهران با همکاری دانشگاه صنعتی شـــریف و انجمن کامپیوتر ایـــران، کنفرانس فنـــاوری اطلاعات در مدیریت شـــهری را در تاریخ ۱۶ اســـفند ۱۳۹۵ در برج میلاد برگزار می‌کند.
زمینه‌های علمی کنفرانس عبارتند از:
۱- حمل و نقل و ترافیک
۲- فرهنگی و اجتماعی
۳- هوشمندسازی و اقتصاد شهری
۴- محیط زیست و سلامت شهری
۵- خدمات شهری
۶- ایمنی و مدیریت بحران
۷- توسعهٔ شهری، شهرسازی و معماری
نشـــانی وبگاه کنفرانس http://csi.org.ir/conf.itcm95 است.

اخبار جهان

همـــکاری اپل با رقیبـــان برای پیشـــبرد پژوهش‌هـــای هوش مصنوعی

شـــرکت اپل برای پیشبرد پژوهش‌ها و تولید و توســـعهٔ فناوری‌هـــای هـــوش مصنوعی به همکاری با شرکت‌های رقیب پرداخته است.
پس از گذشـــت چند ماه از همـــکاری اپل با دیگر بنیان‌گذاران گروه «شـــراکت در هوش مصنوعی» نظیر گوگل، آی‌بی‌ام، مایکروسافت، فیســـبوک و آمازون، شرکت اپل نیز رسماً به این گروه پیوست.
گروه «شراکت در هوش مصنوعی» در سپتامبر سال گذشـــته با هدف هم‌افزایی نیروها برای پیشبرد پژوهش‌های هوش مصنوعی، تأسیس شـــد. عقیدهٔ اعضای این گروه بر این است که هوش مصنوعی می‌تواند در زمینه‌هایی مانند بهداشت، حمل و نقل و خودکارسازی صنایع کمک شایانی کند.
مهم‌ترین فناوری هـــوش مصنوعی اپل که تا کنون عرضه شده، دســـتیار صوتی «سیری» اســـت که می‌تواند به پرســـش‌ها پاسخ دهد. امّا این شـــرکت هنوز در حـــال تبیین راهبرد وســـیع‌تری در زمینهٔ هوش مصنوعی اســـت. مایکروســـافت، فیســـبوک، آی‌بی‌ام، آمازون و گـــوگل راهبردهای خـــود را در این زمینه مشخص نموده‌اند. پیش‌بینی می‌شود که اپل در صدد پیاده‌سازی هوش مصنوعی در پروژهٔ ماشین‌های خودران که می‌توانند بدون دخالت انسان هدایت وسیلهٔ نقلیه را انجام دهند باشد.
آمـــازون افزون بر دســـتیار صوتی الکســـا، از هوش مصنوعی بـــرای ارائهٔ توصیه‌های خرید اســـتفاده می‌کند. گوگل این هفته اعلام کرد که ابزارهای «تنسورفلو» را آماده کرده که به کاربران امکان می‌دهند تا انواع گســـترده‌ای از امکانـــات هوش مصنوعـــی را در رایانه‌های رســـپ‌بری پی‌آی مدل ۳ پیاده‌سازی کنند. آی‌بی‌ام نیز در حال تولید رایانه‌های شناختی می‌باشد.
گـــروه شـــراکت در هوش مصنوعـــی بر این باور اســـت که هوش مصنوعی باعث تأثیرات اجتماعـــی عظیمـــی خواهد گردیـــد و بدین خاطر، این تأثیرات باید مورد بحث قرار گیرند و شـــرکت‌ها باید مقررات خاصی را در مورد چگونگی تولید، توســـعه و بـــه کارگیری این فناوری وضع کنند.
گروه شراکت در هوش مصنوعی هیئت مدیرهٔ متنوعی از افرادی بـــا دیدگاه‌های مختلف از شرکت‌های عضو، دانشـــگاه‌ها و سازمان‌هایی مانند اتحادیهٔ آزادی‌های اجتماعی، مؤسســـهٔ اقتصاد بین‌الملل و بنیاد مک آرتور دارد.

اینفو ورلد، ۲۷ ژانویه ۲۰۱۷

توقف پشـــتیبانی از برنامک‌های ۳۲ بیتی توسط اپل

شـــرکت اپـــل اعلام کـــرد که پشـــتیبانی از برنامک‌های (app) ۳۲ بیتی را متوقف می‌کند و این‌گونه برنامک‌ها دیگر با نسخه‌های آیندهٔ سیستم عامل iOS کار نخواهند کرد. این پیام هم‌اکنون به هنگام راه‌اندازی برنامک‌های ۳۲ بیتی، به صورت یک هشدار بالاپر (pop-up) به معرض دید کاربر گذاشته می‌شود.
شرکت اپل در اکتبر ۲۰۱۴ به تولیدکنندگان برنامک‌هـــای جدیدی که پـــس از تاریخ اول فوریهٔ ۲۰۱۵ نوشته می‌شوند اعلام کرده بود که بایـــد از ۶۴ بیتی پشـــتیبانی کنند. کوته زمانی پـــس از آن، اپل اعلام کـــرد که تمام بروز رسانی‌های برنامک‌ها نیز باید با ۶۴ بیتی سازگار باشند. تمام برنامک‌های ۳۲ بیتی که پس از جون ۲۰۱۵ برای اپل فرستاده شدند، مورد پذیرش قرار نگرفتند. در ماه ســـپتامبر

گذشـــته، اپل اعلام کرد که هر برنامکی که از دســـتورالعمل جدید پیروی نکرده باشد را از فروشگاه اینترنتی خود خارج خواهد کرد.
اپل اعلام نکرده اســـت که کدام نســـخهٔ iOS فقط ۶۴ بیتی خواهد بود امّا این چون تغییر عمده‌ای است احتمالاً در iOS 11 اتفاق خواهد افتاد.
این اقدام بدین معنی اســـت که دستگاه‌های قدیمی‌تر اپل که بر پایـــهٔ معماری ۳۲ بیتی ساخته شده‌اند، قادر نخواهند بود به سیستم عامل جدید iOS ارتقاء یابند. این دســـتگاه‌ها شـــامل آیفـــون 5C و آیفون‌هـــای قدیمی‌تر، نسخهٔ استاندارد آی‌پد و نخستین آی‌پد مینی می‌شود.

اینفو ورلد، ۳۱ ژانویهٔ ۲۰۱۷

تراشه‌های چینی

تعداد تراشه‌های قدرتمند چینی رو به افزایش اســـت. این در حالی اســـت که جنگ لفظی دربارهٔ نیمـــه هادی‌ها هم با آمریکا بالا گرفته است.
شـــرکت فناوری نیمه هادی هواژین‌تونگ که در ایالت گیژو چین با همکاری شرکت کلکام تأسیس شده، تولید تراشهٔ جدیدی را بر پایهٔ فناوری ARM آغاز کرده اســـت. این تراشه‌ها بـــرای بازار چین طراحی شـــده‌اند. شـــرکت AMD نیز همکاری مشـــترکی را با چینی‌ها برای تولید تراشه‌های کارســـاز x86 (server) آغاز کرده است.
علّت رو آوردن تراشه سازها به بازار چین این است که فرصت‌های زیادی برای فناوری‌های مراکز داده در آن کشور وجود دارد. شرکت‌های چینی علی بابا و تِن‌سنت، همانند فیسبوک و گوگل در آمریکا، در حال ســـاخت مراکز دادهٔ عظیمی برای خدمات ابری و یادگیری ماشین هستند.
امّا بـــازار چین ویژگی‌های خاص خود را دارد زیرا شـــرکت‌های آنجا ترجیـــح می‌دهند تا سخت‌افزار مورد نیازشـــان را از فروشندگان محلّی خریداری کنند. دلیل این امر این است که کارســـازهایی که در چین تولید می‌شوند ارزان‌تر هســـتند و خطرات کمتـــری از نظر امنیت ملی دارند.
هـــدف بلند مدّت چیـــن، خوداتکایی در بازار سخت‌افزار است، به نحوی که کلیهٔ دستگاه‌ها در کشور از قطعات تولید داخل ساخته شوند. این کشـــور هم اکنون ســـریع‌ترین اَبَر رایانهٔ جهان را به نام «تای هولایت» در اختیار دارد. شرکت چینی تســـینگ هوا در حال ساخت کارخانهٔ تراشه‌سازی به ارزش ۳۰ میلیارد دلار است. این شرکت ۴/۳ میلیارد دلار در شهری که کارخانه در آن قرار دارد برای خدمت‌رسانی به کارخانه سرمایه‌گذاری کرده است.
در ســـال ۲۰۱۴، دولت چیـــن اعلام کرد که ۱۵۰ میلیارد دلار ظرف مدّت ۱۰ ســـال برای رشد بازار داخلی نیمه هادی‌ها هزینه خواهد کرد. آمریکا چین را متهم کرده اســـت که با دادن امتیازهای غیرمنصفانه به شـــرکت‌های تراشه‌ساز چینی، در بازار نیمه هادی‌ها تقلّب می‌کند.
هم اکنون ۹۰ درصد بازار تراشه‌های کارساز چین در دست اینتل است ولی کلکام تلاش می‌کند که تراشـــه‌های خـــود را که بر پایهٔ معماری ARM ساخته شده‌اند جلو بیندازد. کلکام در اواخر ســـال گذشـــته، نخستین تراشهٔ کارساز ۴۸ هســـته‌ای خود را به نام ســـنتریک ۲۴۰۰ عرضه کرد. این بهترین تراشـــهٔ کارســـاز ARM اســـت که تا کنون ساخته شده است.
ایالت گیژو بـــه صورت مرکـــزی برای کلان داده‌هـــا در چیـــن درآمـــده و بســـیاری از شـــرکت‌های ارتباطـــات راه دور و فراهـــم کنندگان خدمات ابری، مراکز داده‌هایشان را در این ایالت تأسیس کرده‌اند. تولید تراشه‌ها و کارســـازهای محلّی باعث رونق اقتصاد این منطقه و اشتغال‌زایی خواهد شد.

آی دی‌جی نیوز، ۳۰ ژانویه ۲۰۱۶

پیشتازی اپل در بازار تلفن‌های هوشمند

شـــرکت اپل با عرضهٔ تلفن‌هـــای آیفون ۷ و ۷ پـــلاس دوباره مـــکان نخســـت را در بازار تلفن‌هـــای هوشـــمند از آن خود کـــرد. این در حالی اســـت که رقیب اصلـــی اپل یعنی سامسونگ هنوز دســـت به گریبان فراخوانی تلفن‌های گالکســـی نوت ۷ به خاطر گرمای شدید باتری‌هایش است.
شـــرکت اپل در سه ماههٔ چهارم سال ۲۰۱۶، ۷۸/۳ میلیون تلفن هوشمند به فروش رسانده که ۱۷/۸ درصد ســـهم بازار را تشـــکیل داده اســـت. در همین مـــدّت ۷۷/۵ میلیون تلفن هوشـــمند سامســـونگ به فروش رسیده که ۱۷/۷ درصد سهم بازار بوده است.
البته در کلّ سال ۲۰۱۶، سامسونگ با فروش ۳۰۹/۴ میلیون تلفن هوشمند برتری آشکاری نسبت به اپل داشـــته است. فروش تلفن‌های هوشمند اپل در سال گذشته ۲۱۵/۴ میلیون دستگاه بوده است.
بنابر گزارش‌ها، فروش کل تلفن‌های هوشمند در سه ماههٔ چهارم ســـال ۲۰۱۶ با ۹ درصد افزایش به ۴۳۸/۷ میلیون دستگاه بالغ گشته اســـت. رشـــد فروش برای کلّ ســـال ۲۰۱۶ معادل ۳/۳ درصد بوده است. در سال ۲۰۱۶ جمعاً ۱/۵ میلیارد تلفن هوشـــمند به فروش رسیده است.
پس از اپل و سامســـونگ، سه شرکت چینی هوآوی، اوپو و ویوو قرار گرفته‌اند. هوآوی ۱۰ درصد سهم بازار را در سه ماههٔ چهارم ۲۰۱۶ به دست آورده است. این نخستین باری است که سهم بازار هوآوی دو رقمی شده است.

آی‌تی‌ورلد، ۳۱ ژانویهٔ ۲۰۱۷

درخواســـت مایکروسافت برای پیمان جهانی علیه جنگ رایانه‌ای

برد اســـمیت، رئیـــس مایکروســـافت، در وب‌نوشـــت (blog) خـــود خواهـــان پیمان (کنوانســـیون) ژنـــو برای مقابلـــه با جنگ رایانـــه‌ای شـــد. او علّت این درخواســـت را افزایـــش تنش‌های جهانی بر ســـر حملات دیجیتالی عنوان کرده اســـت. مایکروسافت خواهان آن اســـت که اســـتفادهٔ شهروندان از اینترنـــت، به‌عنوان بخشـــی از یک توافق جهانی، مورد حفاظت قرار گیرد. او همچنین

خواستار اتحاد صنایع فناوری برای محافظت از کاربران شد.

بـه عقیدهٔ او چنیـن پیمانی ضـرورت دارد زیرا جنگ رایانه‌ای، زیرسـاخت‌هایی را در بر می‌گیرد که توسط شرکت‌های خصوصی نظیر مایکروسافت کنترل و اداره می‌شوند. به‌علاوه، برخی از حملات مانند رخنه به سیستم‌های رایانه‌ای سـونی در سـال ۲۰۱۴ که به کرهٔ شمالی نسبت داده شـد، شهروندان عادی را هدف قرار می‌دهند.

به گفتهٔ اسمیت، امروز بخش فناوری به‌عنوان نخسـتین واکنشـگر به حملاتی که توسط کشـورها به اینترنت صـورت می‌گیرد، عمل می‌کند. حملهٔ رایانه‌ای توسط یک کشور ابتدا با واکنش و پاسـخی از سوی یک کشور دیگر روبرو نمی‌شود بلکه این شهروندان غیردولتی هستند که پاسخگو می‌باشند.

اشـارهٔ اسـمیت بـه حملـه‌ای اسـت که مایکروسافت در سـال گذشته درگیرش بود، هنگامی که کشـف شد یک عامل کشوری از نام دامنه‌هایی شـبیه نام تجاری این شرکت اسـتفاده می‌کند. مایکروسافت بعداً به حکم دادگاه اجازه یافت که ترافیک گسیل شده به سوی آن دامنه‌ها را تغییر مسیر دهد و حمله را دفع کند.

مایکروسافت از تابسـتان گذشته تاکنون در واکنش به چنین حمله‌ای، ۶۰ نام دامنه را در ۴۹ کشور در ۶ قارهٔ جهان غیرفعّال کرده است.

آی‌تی‌ورلد، ۱۴ فوریهٔ ۲۰۱۷

عرضهٔ تراشـهٔ کیتسون توسط اینتل

شرکت اینتل پس از گذشت بیش از سه سال، گونهٔ بعدی و احتمـالاً آخرین گونهٔ پردازندهٔ ایتانیوم خـود را به نام کیتسـون روانهٔ بازار می‌کند.

این تراشه اکنون در اختیار مشتریان آزمایشی گذاشته شده و فروش انبوه آن در ماه‌های بعد صورت خواهد گرفت.

تراشـه‌های ایتانیوم در رایانه‌هـای بزرگ و کارسـازهایی با وظایف حساس به کار گرفته شده‌اند. شرکت هیولت پاکاردانترپرایز (HPE) اعلام کرده اسـت که کارسازهایی را با تراشهٔ کیتسون در سال جاری عرضه خواهد کرد.

تراشـه‌های ایتانیـوم روزهای آخرشـان را می‌گذرانند و کیتسـون احتمالاً آخر خط آن خواهد بود. پشـتیبانی از تراشه‌های ایتانیوم کاهش یافته و تولید نرم‌افزار برای آن متوقف گشـته است. اینتل به‌طور آشـکار مشتریان را تشـویق به انتقال از ایتانیوم به تراشه‌های زیون می‌کند. این تراشـه‌ها ۹۰ درصد سهم بازار تراشه‌های کارسازها را در اختیار دارند.

اینتل تراشـهٔ ایتانیوم را در سـال ۲۰۰۱ با همکاری شرکت هیولت پاکارد تولید و عرضه کرد. هدف از عرضهٔ این تراشـهٔ بسیار سریع، جایگزینی معماری‌هـای قدیمی‌تر رایانه‌های بزرگ نظیر اسـپارک، و رقابت با تراشهٔ پاور آی‌بی‌ام بود. امّا ایتانیوم به این هدف دسـت نیافت و نتوانسـت اسپارک یا پاور را از میدان به در کند. بزرگ‌ترین دشمن ایتانیوم، تراشهٔ دیگر خود اینتل به نام زیون بود که با موفقیت در بازار رو به رشـد کارسـازهای متوسط و کوچک جذب شـد. به دنبال آن، اینتل منابع بیشتری را در تولید زیون به کار گرفت.

مرگ تدریجی ایتانیوم هنگامی بیشـتر جلب توجـه کرد که اوراکل در سـال ۲۰۱۱ اعلام کرد که تصمیم گرفته نوشـتن نرم‌افزار برای آن معماری را متوقف سازد زیرا به نظر می‌آید که این تراشـه به پایان عمرش رسیده است. مایکروسـافت نیـز پـس از اوراکل از تولید نرم‌افزار برای ایتانیوم دست کشید.

تنها فروشندهٔ عمده‌ای که هنوز از تراشه‌های ایتانیوم در کارسـازهایش اسـتفاده می‌کند HPE اسـت کـه از آن‌ها در کارسـازهای با وظایف حساس استفاده می‌کند. این شرکت به مشتریانش اطمینان داده که تا سال ۲۰۲۵ به پشتیبانی از ایتانیوم ادامه خواهد داد.

پی‌سی‌ورلد، ۱۴ فوریهٔ ۲۰۱۷

مرکز آی‌بی‌ام بـرای رایانه‌های پوشیدنی

پژوهشگران در شرکت آی‌بی‌ام مرکزی را برای رایانه‌های پوشیدنی ایجاد کرده‌اند که می‌تواند اطلاعـات را از چندین دسـتگاه پوشـیدنی جمع‌آوری کند و آن‌ها را با پزشـک در میان بگذارد و به‌طور بالقوّه باعث کاهش زمانی که بیمار نیاز دارد در بیمارستان بگذراند گردد.

ایـن ابـزارک (gadget) که آی‌بـی‌ام آن را «راهنمای شـناختی» نام نهاده، داده‌ها را از دسـتگاه‌هایی نظیر سـاعت‌های هوشمند و نوارهای برازشی (fitness) جمع کرده و به ابر آی‌بی‌ام می‌فرستد. داده‌ها در آنجا تحلیل شده و نتایج به آگاهی کاربر و پزشک او می‌رسد.

ایدهٔ اصلی این اسـت که بیمـاران می‌توانند بدین طریق به طور مطمئنی مورد پایش قرار گیرند و بنابراین می‌توان آن‌ها را یک یا دو روز زودتر از بیمارسـتان به خانه فرستاد. و دیگر این که در صورت پیش‌آمدن مشکل، پزشک بلافاصلـه آگاهی می‌یابد و چنانچه مشـکل جدّی باشـد آمبولانس به نشانی بیمار اعزام خواهد شد.

آی‌تی‌ورلد، ۱۴ فوریهٔ ۲۰۱۷

اضافه شدن آوت لوک به فهرست خدمات پولی مایکروسافت

مایکروسافت رسماً سرویس پست الکترونیکی Outlook.com رادر فهرسـت خدمـات پولی خود قرار داد. این سـرویس که حق اشتراک سالانه‌اش ۵۰ دلار اسـت، تبلیغات را حذف کـرده، پنج صندوق پسـتی فراهم کرده و از نشانی‌های شخصی پشتیبانی می‌کند.

Outlook.com کسـب و کارهـای کوچک و خانواده‌هایی با ۵ حسـاب کاربری و نشـانی شـخصی را هدف قرار داده اسـت. نام دامنه برای مورد اخیرالذکر برای سـال اول رایگان می‌باشـد و کسانی که نام دامنه داشته باشند می‌توانند از آن به‌طور رایگان همراه با سرویس پست الکترونیکی استفاده کنند.

کسـانی که تـا پایان مـاه مارچ نسـبت به خریداری این سـرویس اقدام کنند، ۳۰ دلار تخفیف دریافت خواهند کرد.

کامپیوتر ورلد، ۱۴ فوریهٔ ۲۰۱۷

نسخهٔ جدید زبان GO

آخرین نسخهٔ زبان محبوب و متن‌باز گوگل به نام GO 1.8 دردسترس قرار گرفته است. تمرکز نسخهٔ ۱/۸ بر ترجمه (کامپایل)، زباله‌روبی و پشتیبانی HTTP بوده است. بهینه‌سازی برنامهٔ مترجم ابتدا در نسخهٔ ۱/۷ زبان GO (ماه آگوست گذشته) برای پردازنده‌های ۶۴ بیتی X86 صورت گرفته بود امّا اکنون در تمام معماری‌ها مورد استفاده قرار می‌گیرد و به گفتهٔ سرپرست تیم GO، بهبود چشمگیری را در عملکرد به همراه خواهد داشت. برای نمونه، زمان مورد نیاز CPU بر روی سیستم‌های ۳۲ بیتی ARM بین ۲۰ تا ۳۰ درصد کاهش خواهد یافت.

زمان لازم برای زباله‌روبی نیز در نسخهٔ ۱/۸ به نحو قابل ملاحظه‌ای کوتاه‌تر شده و معمولاً کمتر از ۱۰۰ میلی ثانیه و گاه حتی به ۱۰ میلی ثانیه می‌رسد.

محبوبیت زبان GO دربین برنامه‌نویسان رو به افزایش است و این زبان در سال ۲۰۱۶ با ۹۳٪ رشد از نظر تعداد برنامه‌نویسان، به رتبهٔ هشتم محبوب‌ترین زبان‌ها صعود کرده است.

اینفو ورلد، ۱۷ فوریهٔ ۲۰۱۷

تولید تلفن‌های آیفون در هند

شرکت اپل در ماه‌های آینده مونتاژ تلفن‌های آیفون مدل SE خود را در شهر بنگلور کشور هند آغاز خواهد کرد.

اپل با این اقدام در صدد بالابردن سهم خود در بازار تلفن‌های همراه هند که سریع‌ترین رشد را در جهان دارد است. بیشترین سهم در بازار تلفن‌های همراه هند متعلّق به تلفن‌هایی است که ارزان‌تر از تلفن‌های آیفون می‌باشند. اپل برای پایین آوردن قیمت، در صدد تولید محلّی است و در حال مذاکره و با مسئولان دولت هند برای استفاده از امتیازهای مالیاتی است.

بنابر گزارش‌ها، اپل برای مونتاژ ۳۰۰ تا ۴۰۰ هزار آیفون SE در هند برنامه‌ریزی کرده است. هنوز مشخص نیست که اپل چه مدل‌های دیگری از تلفن آیفون را در آینده در این کشور مونتاژ خواهد کرد.

اپل در سال گذشته ۲/۵ میلیون تلفن آیفون در هند به فروش رسانده است و در این بازار رتبهٔ دهم را داشته است. سامسونگ و چند شرکت چینی مانند زیائومی و ویوو بازیگران اصلی در بازار تلفن‌های همراه هند هستند و اغلب تلفن‌های آن‌ها قیمتی کمتر از ۱۵۰۰۰ روپیه (۲۲۵ دلار) دارند. این در حالی است که قیمت پایهٔ فروش آیفون مدل SE در فروشگاه اینترنتی آمازون ۴۲۴ دلار است.

رویترز، ۱۷ فوریهٔ ۲۰۱۷

رشد بازار هوش تجاری در سال ۲۰۱۷

بنابر پیش‌بینی مؤسسهٔ گارتنر، بازار نرم‌افزارهای تحلیلی و هوش تجاری (BI) در سال ۲۰۱۷ با ۷/۳ درصد رشد روبرو خواهد شد و به ۱۸/۳ میلیارد دلار بالغ خواهد گشت. بر پایهٔ گزارش گارتنر، شرکت‌های مایکروسافت، تابلو و کلیک پیشتازان این بازار هستند. در این گزارش همچنین آمده است که یادگیری ماشین و خودکارسازی کلّ جریان کار تحلیل، فرصت‌های خوبی را هم برای شرکت‌های نوپا و هم تولیدکنندگان سنتی نرم‌افزارهای هوش تجاری پدید آورده است.

رویترز، ۱۷ فوریهٔ ۲۰۱۷

کاهش فروش محصولات سیسکو

فروش مسیریاب‌های شبکه‌ای سیسکو در سه ماههٔ گذشته با ۱۰٪ کاهش روبرو بوده و فروش کلّی این شرکت نیز نسبت به دورهٔ مشابه سال گذشته ۲٪ کاهش داشته است. فروش محصولات سیسکو در این سه ماه ۱۱/۶ میلیارد دلار بوده است.

فروش مسیریاب‌ها، سوئیچ‌ها و محصولات مراکز داده که کسب و کار سنّتی سیسکو را تشکیل می‌دهند به ترتیب ۱۰٪، ۵٪ و ۴٪ کاهش داشته امّا فروش محصولات امنیتی و بیسیم سیسکو به ترتیب ۱۴٪ و ۳٪ رشد داشته است. فروش خدمات سیسکو نیز با ۵٪ رشد به ۶/۱ میلیارد دلار رسیده است.

کامپیوتر ویکلی، ۱۶ فوریهٔ ۲۰۱۷

اتصال مردم جهان به یکدیگر از طریق فیسبوک

اکنون نزدیک به ۱۳ سال است که فیسبوک تلاش کرده است تا دوستان را به یکدیگر متصل کند. اینک مارک زاکربرگ، مؤسس فیسبوک، برنامهٔ تازه و بزرگ‌تری در سر دارد. او می‌گوید فیسبوک بر ایجاد جامعهٔ جهانی تمرکز خواهد کرد.

به گفتهٔ زاکربرگ، اکنون زمانی است که بسیاری از ما در سرتاسر جهان در اندیشهٔ این هستیم که چگونه می‌توانیم بیشترین تأثیر مثبت را داشته باشیم. ممکن است ما توانایی ایجاد دنیایی که می‌خواهیم را در کوتاه مدّت نداشته باشیم امّا می‌توانیم از همین امروز کار بلند مدت را آغاز کنیم. در زمان‌هایی مانند این، مهم‌ترین کاری که ما در فیسبوک می‌توانیم بکنیم، ایجاد و توسعهٔ زیرساختی اجتماعی است که به مردم توانایی ایجاد یک جامعهٔ جهانی که برای همهٔ ما کارگشا باشد را بدهد.

زاکربرگ می‌گوید در این راه، او می‌خواهد بزرگ‌ترین شبکهٔ اجتماعی جهان به مردم کمک کند تا جوامع پشتیبان بنا کنند، جوامعی ایجاد کنند که در زمان بحران‌ها کمک‌رسان باشند و مردم را تشویق کنند که آگاه‌تر شوند و آگاهانه‌تر رأی دهند.

زاکربرگ این مطلب را در وب‌نوشت خود منتشر کرده است. او گفته است اکنون ایجاد چنین جوامعی از آن جهت اهمیت دو چندان یافته است که بسیاری از مردم از قافله جهانی‌شدن، عقب مانده‌اند.

آی‌تی‌ورلد، ۱۷ فوریهٔ ۲۰۱۷

الگوریتمی مبتنی بر بسته‌بندی بسته‌ها برای مدیریت بهتر منابع در محیط ابر*

زهرا موحدی‌نیا
دانشجوی دکتری، گروه مهندسی معماری کامپیوتر، دانشکده مهندسی کامپیوتر، دانشگاه اصفهان
پست الکترونیکی: z.movahedinia@eng.ui.ac.ir

محمدرضا خیام‌باشی
دانشیار دانشکده مهندسی کامپیوتر، دانشگاه اصفهان
پست الکترونیکی: m.r.khayyambashi@eng.ui.ac.ir

چکیده

کاهش انرژی مصرفی مراکز داده از موضوعاتی است که در سال‌های اخیر مورد توجه بسیاری قرار گرفته است. به این منظور ارائه‌کنندگان ساختار به عنوان سرویس در محیط ابر سعی می‌کنند تا حد امکان در مصرف انرژی صرفه‌جویی و کارسازها** را به حالت خواب ببرند. با استفاده از الگوریتم‌های مبتنی بر بسته‌بندی بسته‌ها می‌توان منابع ابر را به گونه‌ای بهینه، اختصاص و از این طریق مصرف انرژی را کاهش داد. تقسیم منابع مرکز داده با ایجاد ماشین‌های مجازی روی کارسازها و اختصاص آن‌ها به کاربران مختلف ابر ممکن می‌شود. در این پژوهش، با کمک الگوریتم بدترین انطباق کاهشی، ماشین‌های مجازی روی تعداد کمتری از کارسازهای فیزیکی قرار می‌گیرند. به این ترتیب تعداد بیشتری از کارسازها می‌توانند بیکار و خوابیده شوند و انرژی کمتری مصرف کنند. با شبیه‌سازی شبکهٔ ابر و استفاده از نشانگرهای استاندارد، ارزیابی عملکرد روش پیشنهادی انجام گرفته و نشان داده شده است که با این شیوه انرژی مصرفی به میزان قابل توجهی کاهش می‌یابد.

کلمات کلیدی: رایانش ابری، مجازی‌سازی کارساز، الگوریتم‌های بسته‌بندی بسته‌ها.

۱- مقدمه

در پنج سال گذشته مصرف انرژی در مرکز داده‌ها دو برابر شده و با توجه به مضرات زیست‌محیطی این مسئله، کنترل مصرف انرژی امری حیاتی است [۱]. از سوی دیگر با گرایش روزافزون به بهره‌گیری از امکانات رایانش ابری، مدیریت منابع ابر برای سرویس‌دهی به تمامی مشتریان و برآورده کردن انتظارات آن‌ها، برای ارائه‌دهندگان ابر بخصوص ارائه‌دهندگان زیرساخت به عنوان سرویس (IaaS) اهمیت فراوانی یافته است. در یک ابر IaaS منابع سخت‌افزاری مانند پردازنده، حافظه و

* این مقاله در بیست و یکمین کنفرانس ملی سالانهٔ انجمن کامپیوتر ایران، ۱۸-۲۰ اسفند ۱۳۹۴ ارائه شده است.
** Server

دستگاه‌های ورودی/خروجی باید میان میلیون‌ها مشتری تقسیم شود. آنچه این مسئله را ممکن می‌سازد مجازی‌سازی کارسازهای مرکزداده می‌باشد. با ایجاد ماشین‌های مجازی ((Virtual Machine (VM) بر روی کارسازها و اختصاص آن‌ها به مشتریان مختلف می‌توان از منابع یک مرکز داده به میزان بالاتری بهره برد و در عین حال با رعایت توافق سطح سرویس (SLA) انتظارات مشتریان را برآورده نمود. از جمله فواید مجازی‌سازی کارسازها در مرکز داده‌ها موارد زیر می‌باشند [۲]:

• تقسیم منابع بزرگ: زمانی که یک منبع فیزیکی برای یک مشتری زیاد است با مجازی‌سازی می‌توان آن را تقسیم و میان چند کاربر به اشتراک گذاشت.

• تجمیع منابع کوچک: زمانی که یک منبع فیزیکی برای یک مشتری کافی نیست با مجازی‌سازی می‌توان قابلیت‌های چند منبع فیزیکی را برای آن کاربر کنار هم گرد آورد.

• مجزا کردن محیط کار مشتری‌ها نسبت به هم: با مجازی‌سازی، محیط کاربران ابر را می‌توان به گونه‌ای از هم جدا کرد که هیچ تداخل و رابطه‌ای با هم نداشته باشند.

• مدیریت آسان‌تر منابع: به دلیل نرم‌افزاری بودن منابع مجازی، مدیریت آن‌ها آسان‌تر است.

برای کاهش میزان مصرف انرژی با کمک روش‌های موسوم به بسته‌بندی بسته‌ها[1] حداکثر تعداد ماشین‌های مجازی ممکن روی هر کارساز قرار می‌گیرد و به این ترتیب کارسازها تا حد امکان خالی و در نهایت کارسازهای بیکار، به حالت خواب می‌روند. نشان داده شده است که انرژی مصرفی یک کارساز خوابیده حداکثر برابر با ده درصد از انرژی مصرفی یک کارساز فعال است. البته، یک کارساز خواب برای فعال شدن تنها به چند میلی‌ثانیه زمان احتیاج دارد [۳]. با توجه به این‌که این میزان زمان در بسیاری از کاربردها قابل چشم‌پوشی است، خواباندن کارسازهای بیکار راه‌حل مناسبی برای کاهش مصرف انرژی مرکزداده می‌باشد.

در این مقاله بر منبع پردازنده در کارسازها تمرکز شده است و با استفاده از بدترین برازش کاهشی که یکی از الگوریتم‌های بسته‌بندی بسته‌ها محسوب می‌شود، آرایش ماشین‌های مجازی مشتری‌ها روی کارسازها تنظیم می‌گردد. در ادامه این مقاله، در بخش دوم مروری بر پژوهش‌های پیشین و در بخش سوم الگوریتم WFD توضیح داده شده است. در انتها نیز در بخش چهارم عملکرد سیستم پیشنهاد شده مورد ارزیابی قرار می‌گیرد و سپس در بخش پنجم، جمع‌بندی مقاله، نتیجه‌گیری و راه‌کارهای آینده ارائه می‌گردد.

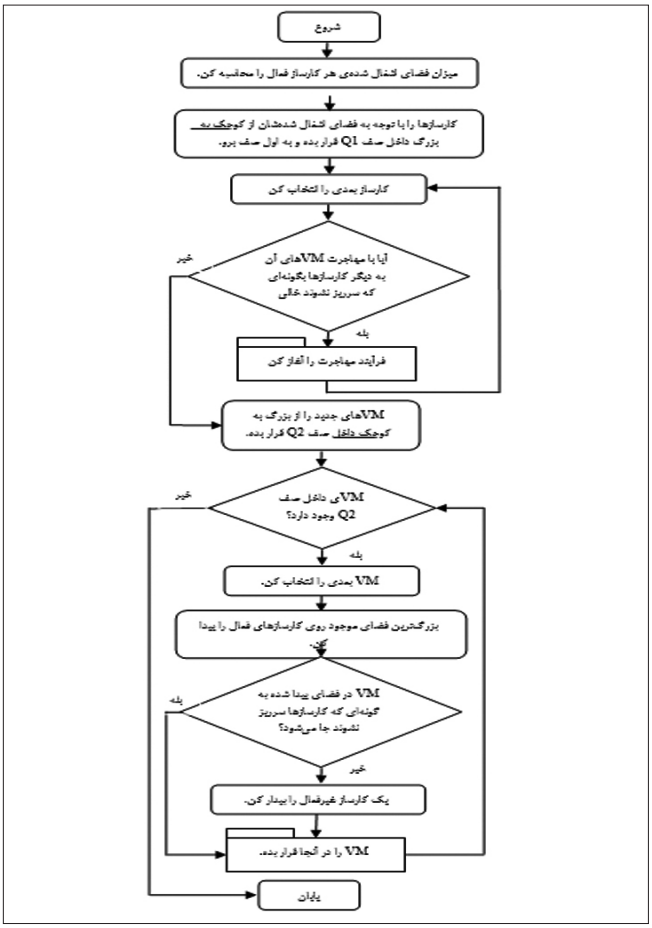
۲- مرور بر پژوهش‌های پیشین

در مطالعات پیشین راه‌حل‌های مختلفی برای کاهش تعداد کارسازهای فعال مرکز داده پیشنهاد شده است. در مقالهٔ [۴] روشی متشکل از دو فاز پیش راندن و به عقب کشیدن ارائه گردیده است که در فاز اول آنقدر منابع به کاربران اختصاص داده می‌شود تا بی‌نیاز شوند. سپس در فاز دوم تا حد ممکن منابع اضافی از مشتریان باز پس گرفته می‌شود. در تحقیق [۵] روشی به نام اتوکنترل پیشنهاد شده است که در آن هر سرویس توسط برداری شامل سه عضو (هدف، پارامتر و اولویت) توصیف می‌گردد. اولویت بیانگر ارجحیت برنامه، پارامتر نشان‌دهندهٔ میزان بهره‌وری سیستم و هدف معرف میزان مطلوب بهره‌وری سیستم است. بهره‌وری سرویس‌ها برای منابع پردازنده و وسایل ورودی/خروجی محاسبه می‌شود. کنترل‌کننده با استفاده از حسگر خود، بهره‌وری سرویس‌ها را محاسبه و با میزان مورد نظر (هدف) مقایسه می‌نماید و بر این اساس در مورد نحوهٔ تخصیص منابع تصمیم می‌گیرد. در مقاله‌های [۳ و ۶] نشان داده شده است که برای بالا بردن ستبری روش تخصیص منابع و پیشگیری از سرریز شدن کارسازها، نباید کارسازها کاملا پر و از تمامی ظرفیت آن‌ها استفاده شود. همچنین در این مقاله اندازهٔ VMها متغیر فرض شده است. رسالهٔ [۷] بر استفاده بهتر از منابع کارسازها مانند پردازنده و حافظه درکنار پهنای باند تمرکز کرده و به این منظور VMهایی که لازم است با هم ارتباط داشته باشند را در کنار هم و حتی‌الامکان روی یک کارساز قرار داده است. در مقاله [۸] به VMها با دید وظایفی نگاه شده است که باید تا قبل از یک ضرب‌الاجل مشخصی انجام گیرند. با تمام شدن یک وظیفه، در صورت وجود فضای کافی روی کارساز، وظیفه دیگری می‌تواند به آن کارساز مهاجرت کند و جای او را بگیرد. در این مقاله با زمان‌بندی وظایف (VMها) سعی می‌شود استفاده از منابع کارسازها بهینه شود. در تحقیق [۹] به مسئلهٔ نامتجانس بودن نیازمندی‌های VMها توجه شده است. ممکن است یک VM به محاسبات سنگین احتیاج داشته باشد و در اختیار گرفتن یک پردازندهٔ خوب برایش بحرانی باشد در حالی که برای VM دیگر داشتن یک حافظهٔ پرسرعت و بزرگ مهم باشد. در این مقاله گفته شده است که VMهایی که نیازمندی‌های متفاوتی دارند باید کنار هم و روی یک کارساز قرار گیرند. اگر VMهایی که نیازمندی‌های مشابهی دارند با هم باشند آنگاه آن منبع به یک گلوگاه تبدیل می‌شود. به این ترتیب همبستگی (Correlation) میان VMها روی منابع مختلف حساب می‌شود و آن‌هایی که همبستگی کمتری دارند برای کنار هم بودن انتخاب می‌گردند.

۳- الگوریتم پیشنهادی

در الگوریتم‌های مبتنی بر بسته‌بندی بسته‌ها فرض می‌شود عناصری با اندازه‌های مختلف وجود دارند که باید به گونه‌ای داخل بسته‌هایی با اندازه‌های مشخص قرار داده شوند که کمترین تعداد بسته مصرف و کمترین فضا هدر رود. به عبارت دیگر، بیشترین

1-Bin Packing Algorithm



شکل ۱: فلوچارت الگوریتم قرار دادن VMها روی کارسازها

۱. شروع
فاز اول:
۲. قرار بده **s** = کارسازهای فعال
۳. قرار بده **n** = اندازه **s**
۴. قرار بده **i** = ۰
۵. قرار بده **Q1[i]** = المان یا بیشترین فضای آزاد از **s**
۶. عنصری یا بیشترین فضای آزاد را از **s** حذف کن.
۷. قرار بده **i** = **i** + ۱
۸. اگر **i** < **n** آنگاه برو به ۵
۹. قرار بده **i** = ۰
۱۰. اگر امکان‌مهاجرت تمامی**vm**های**Q1[i]**وجود ندارد آنگاه برو به ۲۰
۱۱. قرار بده **v1** = **vm**های **Q1[i]**
۱۲. قرار بده **m** = اندازه **v1**
۱۳. قرار بده **j** = ۰
۱۴. **v1[j]** را به **Q1[i + ۱]** مهاجرت بده.
۱۵. جای المان **i + ۱** ام از **Q1** را به روز کن.
۱۶. قرار بده **j** = **j** + ۱
۱۷. اگر **m** > **j** آنگاه برو به ۱۴
۱۸. قرار بده **i** = **i** + ۱
۱۹. اگر **n** > **i** آنگاه برو به ۱۰
فاز دوم:
۲۰. **vm**های جدید را بگیر و در **v2** قرار بده.
۲۱. قرار بده **m** = اندازه‌ی **v2**
۲۲. قرار بده **i** = ۰
۲۳. قرار بده **Q2[i]** = عنصری یا بیشترین قضای درخواستی از **v2**
۲۴. عنصری یا بیشترین قضای درخواستی را از **v2** حذف کن.
۲۵. قرار بده **i** = **i** + ۱
۲۶. اگر **m** > **i** آنگاه برو به ۲۳
۲۷. قرار بده **i** = ۰
۲۸. قرار بده **s** = کارسازهای فعال
۲۹. قرار بده **h** = عنصری یا بیشترین قضای آزاد از **s**
۳۰. اگر قرار دادن **v2[i]** روی **h** آن را سرویز نمی‌کند آنگاه برو به ۳۳
۳۱. قرار بده **s** = کارسازهای غیرفعال
۳۲. قرار بده **h** = عنصری یا بیشترین قضای آزاد از **s**
۳۳. **v2[i]** را روی **h** قرار بده.
۳۴. قرار بده **i** = **i** + ۱
۳۵. اگر **m** > **i** آنگاه برو به ۲۸
۳۶. پایان

شکل ۲: شبه‌کد الگوریتم قرار دادن VMها روی کارسازها

تعداد عناصر در هر بسته قرار گیرد. این مسئله NP-دشوار بوده‌و پیداکردن راه‌حل‌بهینهٔ‌آن‌بسیارزمانبراست،اماروش‌های‌مکاشفه‌ای وجود دارند که می‌توانند نتایجی بسیار نزدیک به پاسخ بهینه ارائه دهند. اولین راه‌حلی که برای حل مسئلهٔ بسته‌بندی بسته‌ها به ذهن می‌رسد این است که هر عنصر در اولین بسته‌ای که جا می‌شود قرار داده شود. به این روش اولین برازش گفته می‌شود. راه‌حل بهتر این است که عناصر از بزرگ به کوچک مرتب و ابتدا عناصر بزرگ‌تر داخل بسته‌ها قرار داده شوند. این روش که با نام اولین برازش کاهشی (FFD) شناخته می‌شود بهتر از روش قبلی است اما باز هم بهینه نیست. فرض کنید تعداد بسته‌ها ثابت است و باید بیشترین تعداد عناصر در هر یک قرار گیرند. پس از مرتب کردن عناصر، در روش بهترین برازش کاهشی (BFD) هر عنصر در کوچک‌ترین فضایی که در آن جا می‌شود قرار و در مقابل، در الگوریتم بدترین برازش کاهشی (WFD) هر عنصر در بزرگ‌ترین فضای ممکن قرار می‌گیرد. در شرایط مختلف، هر یک از الگوریتم‌های BFD و WFD ممکن است بهتر از دیگری عمل کند.

الگوریتم پیشنهادی در این مقاله متشکل از دو فاز می‌باشد. فرض کنید کارسازها به دو دستهٔ فعال و غیرفعال تقسیم شوند. هرگاه درخواست سرویس دادن به یک VM جدید فرارسد، در فاز اول، سعی می‌شود آرایش VMهای قبلی روی کارسازها بهینه گردد. به این منظور ماشین‌های فعال با توجه به میزان فضای اشغال شده از کوچک به بزرگ مرتب و ماشین با کمترین فضای اشغال شده انتخاب و با مهاجرت VMهای آن به دیگر کارسازهای فعال، بیکار و خواب می‌شود. این کار تا جایی ادامه می‌یابد که دیگر کارسازهای فعال برای VMهای کارساز انتخاب شده جا نداشته باشند و کارساز مورد نظر نتواند خالی شود. به‌دلیل این‌که مهاجرت کارسازها پرهزینه و زمان‌بر است اگر تمامی VMهای ماشین مورد نظر قابل مهاجرت به دیگر کارسازها نباشند، اصلا کار مهاجرت آغاز نخواهد شد و هیچ یک از VMها انتقال داده نمی‌شود.

در فاز دوم VMهای جدید با توجه به الگوریتم WFD روی کارسازها قرار داده می‌شوند. با این تفاوت که تا یک کارساز فعال جا دارد، کارسازهای غیرفعال بیدار نمی‌گردند. اگر n عدد VM جدید وجود داشته باشد، پس از مرتب شدن آن‌ها به ترتیب نزولی، بزرگ‌ترین انتخاب و کارساز فعال با بزرگ‌ترین فضای آزاد پیدا می‌شود. اگر VM در آن کارساز جا می‌گرفت که قرار داده می‌شود، در غیر این‌صورت یک کارساز غیرفعال جدید فعال و مقداری از ظرفیت آن را اشغال

جدول ۱: مصرف انرژی و نقض SLA برای سه الگوریتم FFD، BFD و WFD

الگوریتم WFD		الگوریتم FFD		الگوریتم BFD			
نقض SLA	مصرف انرژی kwatt	نقضSLA	مصرف انرژی kwatt	نقض SLA	مصرف انرژی kwatt	تعداد VMها	تعداد وظایف
٪۰/۰۰	۰/۸۱۰	٪۰/۰۰	۱/۰۹۰	٪۰/۰۰	۱/۸۲۷	۶۹	۵۰۰
٪۰/۰۰	۱/۸۹۰	٪۰/۰۰	۲/۵۸۷	٪۰/۰۰	۴/۶۶۹	۱۶۶	۱۰۰۰
٪۰/۰۰	۳/۲۸۵	٪۰/۰۰	۴/۵۴۹	٪۰/۰۰	۸/۲۹۴	۲۹۲	۲۰۰۰
٪۰/۰۰	۴/۳۲۰	٪۰/۰۰	۵/۹۳۱	٪۰/۰۰	۱۰/۷۰۱	۳۸۱	۴۰۰۰
٪۰/۰۰	۴/۹۵۰	٪۰/۰۰	۶/۸۳۸	٪۰/۰۰	۱۲/۳۲۵	۴۳۹	۶۰۰۰
٪۰/۰۰	۸/۶۸۵	٪۰/۰۰	۱۱/۹۹۰	٪۰/۰۰	۲۱/۱۶۹	۷۷۰	۸۰۰۰
٪۰/۰۰	۱۱/۳۴۰	٪۰/۰۰	۱۸/۰۹۴	٪۰/۰۰	۲۵/۲۹۹	۱۰۰۷	۱۰۰۰۰
٪۰/۰۳	۱۳/۱۴۰	٪۰/۰۳	۱۸/۲۰۳	٪۰/۰۳	۲۸/۰۹۹	۱۱۶۸	۱۲۰۰۰
٪۰/۰۶	۱۷/۰۱۰	٪۰/۰۶	۲۳/۵۴۴	٪۰/۰۶	۳۴/۱۱۹	۱۵۱۱	۱۶۰۰۰
٪۰/۱۰	۱۹/۸۴۵	٪۰/۱۰	۳۰/۱۹۳	٪۰/۱۰	۳۸/۵۲۹	۱۷۶۴	۲۰۰۰۰
٪۰/۲۱	۴۹/۷۶۴	٪۰/۲۱	۶۶/۰۹۹	٪۰/۲۱	۶۹/۶۱۴	۴۲۴۵	۴۰۰۰۰

می‌گردد. بیدار کردن کارسازهای غیرفعال نیز با توجه به الگوریتم WFD انجام می‌شود. به عبارت دیگر، کارساز خواب با بیشترین ظرفیت برای فعال شدن انتخاب می‌گردد. شکل ۱ فلوچارت و شکل ۲ شبه‌کد این الگوریتم را نشان می‌دهد. اگوریتم پیشنهادی با دو الگوریتم دیگر که در آن‌ها به جای WFD از BFD و در دیگری از FFD استفاده شده مقایسه گردیده است.

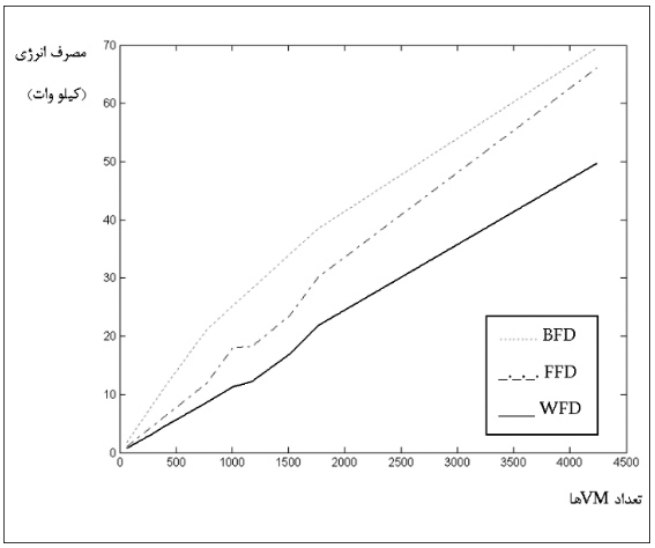
نتایج عملیاتی

برای پیاده‌سازی الگوریتم از CloudSim [۱۰ و ۱۱] استفاده شده است. در مدل مرکز داده، دویست کارساز مختلف شامل شصت‌وشش کارساز تک هسته‌ای با تنظیمات اینتل Core Solo T1xxx، شصت‌وهفت کارساز دو هسته‌ای با تنظیمات اینتل Core i5-45xxxT و شصت‌وهفت کارساز چهار هسته‌ای با تنظیمات اینتل Core i5-46xxxT قرار گرفت و حداکثر ۴۲۴۵ VM با اندازه‌های مختلف روی آن‌ها گذاشته شد. برای ارزیابی، وظایف مورد نیاز شبیه‌سازی از خوشهٔ داده واقعی ابر گوگل [۱۲] تهیه گردید. این داده‌ها در نوامبر ۲۰۱۱ و در طی بیست‌ونه روز از روی خوشه‌ای شامل بیش از دوازده‌هزار کارساز گردآوری شده‌اند. به‌دلیل این‌که حجم این داده‌ها خیلی زیاد است تنها از چهل‌هزار نمونهٔ اول که متعلق به ۱۸۱۳ کاربر مختلف هستند استفاده شده است.

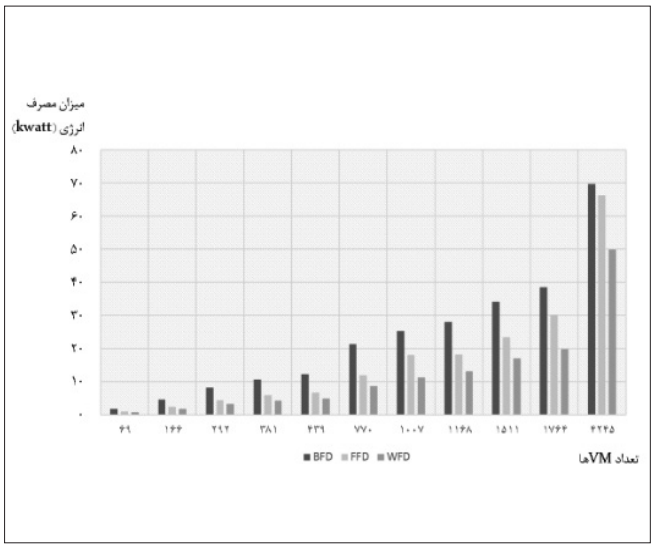
جدول ۱ نتایج شبیه‌سازی را به ازای تعداد VMهای مختلف نشان می‌دهد. شکل ۳ مصرف انرژی در مرکزداده را برای سه الگوریتم BFD، WFD و FFD به تصویر کشیده است. از این میان WFD به دلیل این‌که VMها را روی تعداد کمتری کارساز قرار می‌دهد نتایج بهتری ارائه می‌کند. الگوریتم WFD علاوه بر داشتن مصرف انرژی کمتر، با افزایش تعداد VMها، آن را با شیب کمتری نیز بالا می‌برد. نمودار میله‌ای میزان مصرف انرژی نیز در شکل ۴ نمایش داده شده است. در این برنامه، محاسبات دقیقی برای سرریز نشدن کارسازها انجام شد. به همین خاطر اکثر وظایف به‌موقع پایان یافتند و نقض SLA همان‌گونه که در جدول ۱ قابل مشاهده است، برای هر سه الگوریتم که رفتار مشابهی از این منظر به نمایش گذاشتند، پایین است.

همان‌طور که گفته شد، سربار زمانی حاصل از خواب و بیدار کردن کارسازها ناچیز و قابل چشم پوشی است، اما از آنجا که مهاجرت ماشین‌های مجازی سربار زمانی به همراه دارد، یکی از پارامترهای مهم تعداد مهاجرت‌های صورت گرفته می‌باشد [۱۳ و ۱۴]. جدول ۲ و شکل ۵ و ۶ تعداد مهاجرت‌های انجام شده برای سه الگوریتم FFD، BFD و WFD ونقض SLA در اثر آن را به ازای تعداد VMهای مختلف نمایش می‌دهد.

الگوریتم WFD آرایش ماشین‌های مجازی روی کارسازها را نسبت به دو الگوریتم BFD و FFD متراکم‌تر می‌کند و با کم کردن تعداد کارسازها مصرف انرژی را کاهش می‌دهد. این مسئله موجب می‌شود که تعداد مهاجرت‌ها و متعاقبا نقض SLA کمی بیشتر شود. از جمله راه‌های پیشنهادی برای کاهش سربار حاصل از مهاجرت ماشین‌های مجازی، به اشتراک‌گذاری یک حافظه مرکزی میان کارسازهای مرکز داده و قرار دادن پیکر (image) ماشین‌های مجازی روی آن می‌باشد. به این ترتیب، احتیاجی به مهاجرت کل



شکل ۳: مصرف انرژی برای سه الگوریتم FFD، BFD و WFD



شکل ۴: نمودار میله‌ای مصرف انرژی برای سه الگوریتم FFD، BFD و WFD

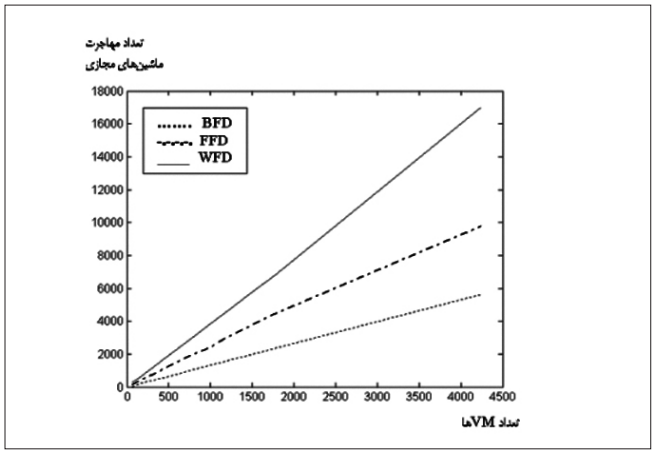
جدول ۲: تعداد مهاجرت‌های ماشین‌های مجازی و نقض SLA در اثر آن برای سه الگوریتم FFD،BFD و WFD

الگوریتم WFD		الگوریتم FFD		الگوریتم BFD			
نقض SLA در اثر مهاجرت	تعداد مهاجرت‌ها	نقض SLA در اثر مهاجرت	تعداد مهاجرت‌ها	نقض SLA در اثر مهاجرت	تعداد مهاجرت‌ها	تعداد VMها	تعداد وظایف
٪۰/۰۰	۲۶۵	٪۰/۰۰	۱۷۴	٪۰/۰۰	۲۹	۶۹	۵۰۰
٪۰/۰۰	۶۳۹	٪۰/۰۰	۴۱۸	٪۰/۰۰	۲۲۰	۱۶۶	۱۰۰۰
٪۰/۰۱	۱۱۲۳	٪۰/۰۰	۷۳۵	٪۰/۰۰	۳۸۷	۲۹۲	۲۰۰۰
٪۰/۰۲	۱۴۶۵	٪۰/۰۰	۹۵۸	٪۰/۰۰	۵۰۵	۳۸۱	۴۰۰۰
٪۰/۰۳	۱۶۸۸	٪۰/۰۱	۱۱۰۴	٪۰/۰۰	۵۶۹	۴۳۹	۶۰۰۰
٪۰/۰۵	۲۹۶۹	٪۰/۰۳	۱۹۳۷	٪۰/۰۱	۱۰۲۱	۷۷۰	۸۰۰۰
٪۰/۰۹	۳۸۷۳	٪۰/۰۶	۲۵۳۳	٪۰/۰۴	۱۳۳۵	۱۰۰۷	۱۰۰۰۰
٪۰/۱۳	۴۴۹۲	٪۰/۰۹	۲۹۳۸	٪۰/۰۶	۱۵۴۸	۱۱۶۸	۱۲۰۰۰
٪۰/۲۲	۵۸۱۱	٪۰/۱۴	۳۸۰۱	٪۰/۱۱	۲۰۰۳	۱۵۱۱	۱۶۰۰۰
٪۰/۲۹	۶۷۸۵	٪۰/۲۲	۴۴۳۷	٪۰/۱۴	۲۳۳۹	۱۷۶۴	۲۰۰۰۰
٪۱/۰۲	۱۷۰۱۸	٪۰/۸۱	۱۰۶۷۷	٪۰/۷۰	۵۶۲۸	۴۲۴۵	۴۰۰۰۰

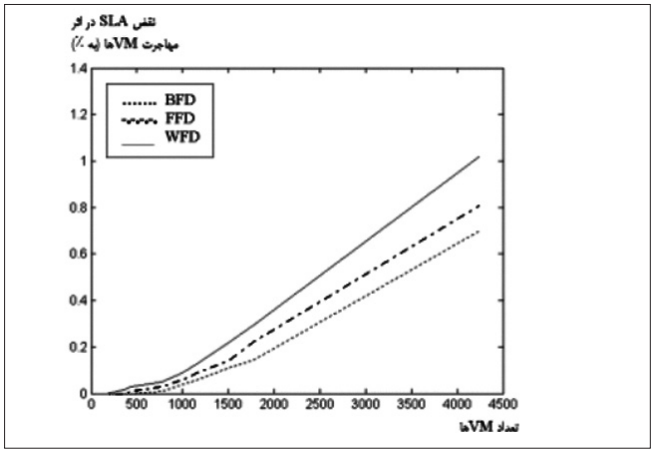
ماشـــین مجازی نخواهد بود و تنها کافی اســـت اطلاعات پردازشی مانند محتویات حافظهٔ تصادفی، حافظهٔ نهان و پشـــته‌ها به کارساز مقصد انتقال یابند. با استفاده از این تکنیک می‌توان هزینهٔ حاصل از مهاجرت‌های بیشـــتر در الگوریتم WFD را کم و قابل چشم‌پوشی کرد [۱۴].

نتیجه‌گیری و راه‌کارهای آینده

در این مقالـــه راه‌حلی برای بهبود بهره‌برداری از منابع کارســـازها و کاهش مصرف انرژی در مراکز دادهٔ ابرهای IaaS پیشـــنهاد گردید. در این روش، با استفاده از الگوریتم WFD به گونه‌ای VMها روی کارسازها قرار داده می‌شـــوند که تعداد کارساز روشن کمتری مورد نیاز باشند.



شکل ۵: تعداد مهاجرت VMها برای سه الگوریتم BFD،FFD و WFD



شکل ۶: نقض SLA در اثر مهاجرت VMها برای سه الگوریتم FFD، BFD و WFD

در این فرآیند سعی می‌شود کارسازها تا حد امکان بیکار و خواب شوند. عملکرد روش پیشنهادی با شبیه‌سازی طرح‌های استاندارد مورد بررسی قرار گرفت و نشان داده شد که مصرف انرژی در سیستم پیشنهاد شده به میزان قابل توجهی کاهش می‌یابد. گرچه تعداد مهاجرت‌های انجام شده در الگوریتم WFD کمی بیشتر از دو الگوریتم دیگر است اما با استفاده از تکنیک قرار دادن پیکر ماشین‌های مجازی روی حافظهٔ مشترک می‌توان سربار زمانی حاصل از این مسئله را کاهش داد و قابل چشم‌پوشی کرد. در ادامه این پژوهش می‌توان علاوه بر قدرت پردازش کارسازها، منابع دیگری مانند حافظه، وسایل ورودی/خروجی و پهنای باند را نیز مورد توجه قرار داد.

مراجع

1- Dong Jiangkang, Wang Hongbo, Li Yangyang, Cheng Shiduan, "Virtual Machine Scheduling for Improving Energy Efficiency in IaaS Cloud," IEEE, Transactions on Energy Conservation and Harvesting for Green Communications, Vol. 11, No. 13, March 2014, PP. 1-12.

2- Raj Jain, Subharthi Paul, "Network Virtualization and Software Defined Networking for Cloud Computing: A Survey," IEEE, Communication Magazine, Vol. 51, No. 11, Nov. 2013, PP. 24-31.

3- Weijia Song, Zhen Xiao, Qi Chen, Haipeng Luo, "Adaptive Resource Provisioning for the Cloud Using Online Bin Packing," IEEE, Transactions on Computers, Vol. 63, No. 11, Nov. 2014, PP. 2647-2660.

4- Nikolaus Huber, Fabian Brosig, Samuel Kounev, "Model-Based Self-Adaptive Resource Allocation in Virtualized Environments," ACM-SEAM '11, ISBN: 978-1-4503-0575-4, 2011, PP. 90-99.

5- Paradeep Padala, Xiaoyun Zhu, et al, "Automated Control of Multiple Virtualized Resources," ACM-EuroSys '09, ISBN: 978-1-60558-482-9, Apr. 2009, PP. 13-26.

6- Lionel Eyraud Dubois, "Hubert Larcheveque, Optimizing Resource Allocation While Handling SLA Violations in Cloud Computing Platforms," IEEE, 27th International Symposium on Parallel & Distributed Processing, ISBN: 978-1-4673-6066-1, 2013, PP. 79-87.

7- J. Dong, X. Jin, H. Wang, Y. Li, P. Zhang, S. Cheng, "Energy-Saving Virtual Machine Placement in Cloud Data Centers," IEEE/ACM, 13th International Symposium on Cluster, Cloud, and Grid Computing, ISBN: 978-1-4673-6465-2, 2013, PP. 618-624.

8- M. Dabbagh, B. Hamdaoui, M. Guizani, A. Rayes, "Release-Time Aware Placement," IEEE-Workshop on Computing Systems, Networks, and Applications, ISBN: 14998988, 2014, PP. 122-126.

9- Yan Zhang, Nirwan Ansari, "Heterogeneity Aware Dominant Resource Assistant Heuristics for Virtual Machine Consolidation," IEEE, ISBN: 14382083, Workshop on Communications QoS, Reliability and Modelling Symposium, 2013, PP. 1297-1302.

10- R. N. Calheiros, R. Rajiv, A. Beloglazov, et al., "CloudSim: a Toolkit for Modelling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms," Wiley, Software: Practice and Experience, Vol. 41, No. 1, June 2010, PP. 23-50.

11- The University of Melbourne, CloudSim: A Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services, updated: 2011, accessed: 2015, http://www.cloudbus.org/cloudsim/.

12- Google, Google Cluster Data, updated: 2014, accessed: 2015, http://code.google.com/p/googleclusterdata/.

13- Andreas Wolke, Martin Bichler, Thomas Setzer, "Planning vs. Dynamic Control: Resource Allocation in Corporate Clouds," IEEE, Transactions on Cloud Computing, Vol. PP, No. 99, Sep. 2014, PP. 1-14.

14- A. Beloglazov, R. Buyya, "Optimal Online Deterministic Algorithms and Adaptive Heuristics for Energy and Performance Efficient Dynamic Consolidation of Virtual Machines in Cloud Data Centers," Wiley, Concurrency and Computation: Practice & Experience, Vol. 24, No. 13, Sep. 2012, PP. 1397-1420.

پیش‌گوی آزمون و کاربرد آن در اشکال‌زدایی نرم‌افزار

علیرضا خلیلیان
دانشجوی دکتری نرم‌افزار گروه مهندسی نرم‌افزار، دانشکده مهندسی کامپیوتر، دانشگاه اصفهان
پست الکترونیکی: khalilian@eng.ui.ac.ir

مریم هاشم‌زاده
دانشجوی کارشناسی ارشد نرم‌افزار گروه مهندسی نرم‌افزار، دانشکده مهندسی کامپیوتر، دانشگاه اصفهان
پست الکترونیکی: m.hashemzade@eng.ui.ac.ir

احمد براآنی دستجردی
دانشیار مهندسی کامپیوتر گروه مهندسی نرم‌افزار، دانشکده مهندسی کامپیوتر، دانشگاه اصفهان
پست الکترونیکی: ahmadb@eng.ui.ac.ir

بهمن زمانی
استادیار مهندسی کامپیوتر گروه مهندسی نرم‌افزار، دانشکده مهندسی کامپیوتر، دانشگاه اصفهان
پست الکترونیکی: zamani@eng.ui.ac.ir

چکیده

شـرکت‌های نرم‌افزاری اکنون به‌درسـتی فهمیده‌اند که برای تضمین کیفیت نرم‌افزار از طریق آزمون و اشکال‌زدایی باید از فنون علمی بهره ببرند و از ابزارهای خودکار اسـتفاده نمایند. یکی از ملزومات این‌کار این اسـت که بدانیم اگر نرم‌افزار بی‌خطا باشـد، موقع آزمون با دادن هر ورودی چه خروجی باید تولید کند. این نیازمندی همان مسئله پیش‌گوی آزمون است که خودکارسازی آن مسئله‌ایست حیاتی و دشوار و مورد نیاز صنعت و پژوهش‌های دانشگاهی. مقاله حاضر به اختصار در مورد این مسئله بحث می‌کند تا ابعاد آن بیشتر روشن گردد.

مقدمه

در بازار روبه رشـد نرم‌افزار، تیم‌های توسـعه نرم‌افزار با چالش‌های بسـیاری مواجهند. متأسفانه، افزایش تقاضا برای نرم‌افزار و رقابت در صنعت نرم‌افزار منجر به کاهش زمان عرضه به بازار و افزایش احتمال عرضه نرم‌افزار معیوب می‌گردد. در نتیجه فعالیت‌های تضمین کیفیت نرم‌افـزار اهمیت فوق‌العـاده پیدا می‌کنند که در ایـن میان آزمون نرم‌افزار در بهبود کیفیـت و قابلیت اطمینان نرم‌افزار نقش حیاتی را ایفا می‌کند.

آزمون نرم‌افزار کاملاً در مقیاس عملی و صنعتی قابل استفاده است و هدفش شناسـایی نواقص موجود در محصول نرم‌افزاری است. مشکل اینجاسـت که آزمون نرم‌افزار فرآیندی زمان‌گیر و پرهزینه است که بیش از ۵۰ درصد از هزینه‌های چرخه حیات توسعه نرم‌افزار را مصرف می‌کند. آزمون نرم‌افزار به دو صورت دستی و خودکار انجام می‌شود. در آزمون دسـتی آزمون‌گران آزمون‌هایی را برای پیدا کردن خطاها و اشـکال‌های نرم‌افزار طراحی می‌کنند. سـپس آزمون‌گران انسانی که برای نرم‌افـزار نقش کاربر نهایی را ایفا می‌کنند، تمام ویژگی‌های برنامـه را به‌کار می‌گیرند و برنامه را اجرا می‌کنند که از رفتار صحیح نرم‌افزار اطمینان حاصل نماید. آزمون دسـتی مشـکلاتی دارد که از عمده‌ترین آن‌ها می‌توان به هزینه و زمان بالا، دشواری و دقت پایین اشاره کرد. خودکارسازی فنون آزمون و اشکال‌زدایی در دو دهه اخیر کمک شـایانی به حل مشکلات مذکور کرده است از این جهت که در مـدت زمان کمتر و با هزینه کمتر می‌تـوان نرم‌افزار را از جنبه‌های بیشـتری آزمود. حاصل چنین آزمون گسـترده‌تر، یافتن نقص‌های بیشتر و افزایش بیشتر کیفیت نرم‌افزار است.

در حال حاضر فنون بسـیار متعددی برای آزمون خودکار وجود دارد. مسـئله این اسـت که با وجود ابزارهای متعدد برای آزمون خودکار، هنـوز آزمون را نمی‌توان کاملاً خودکار کرد زیرا بخشـی از ملزومات آن را نمی‌توان به‌طور خودکار به‌دست آورد. آزمون خودکار به اجرای خودکار و بررسی نتایج به‌صورت خودکار نیاز دارد؛ یعنی فن خودکار بایـد بداند که خروجی منتظـره برنامه در قبال اجـرای هر ورودی چیست تا بتواند خروجی واقعی را با خروجی مورد انتظار مقایسه کند و تشـخیص دهد برنامه اجرای درستی دارد یا خیر. خروجی منتظره بخشی از داده آزمون است.

داده آزمون شـامل ورودی‌های مناسـب و خروجی‌های مورد انتظار اسـت که برای خودکارسـازی آزمون باید داده آزمون را نیز به‌طور خـودکار تولیـد کنیم. ورودی‌های مناسـب داده‌هایی هسـتند که معیارهای خاصی را برآورده سـازند. این معیارها چنان هسـتند که به آزمون‌گران درصدی اطمینان دهند که برنامه در شـرایط مختلف اجرای مطلوبی دارد. بنابراین تولید خودکار ورودی مناسـب یکی از زمینه‌های تحقیقاتی فعال و مؤثر در خودکارسازی آزمون است. برای این منظور فنون بسـیار سـودمند و کارامدی طراحی شده است. اما مشکل فعلی تولید خودکار خروجی‌های مورد انتظار است.

طرح مسئله

مسئله این اسـت که بعد از اجرای برنامه با داده‌های ورودی، چگونه درستی اجرای برنامه را تشخیص دهیم؟ در آزمون دستی، آزمون‌گران افرادی هسـتند که معمـولاً در جریان تحلیل معنایـی نرم‌افزار قرار گرفته‌انـد و با ایـن دانش، خروجـی برنامه را مشـاهده می‌کنند و تشـخیص می‌دهند که درست اسـت یا خیر. اما اگر آزمون خودکار شود، خروجی‌های مورد انتظار هم باید خودکار تولید شود و خودکار مقایسه شوند و حل این مسئله کار ساده‌ای نیست.

در ادبیات آزمون و اشـکال‌زدایی نرم‌افزار بـه خروجی مورد انتظار، پیش‌گو یا اوراکل آزمون می‌گویند. اما در متون و مقالات، پیش‌گوی آزمون به‌طور گسترده‌تری هم استفاده می‌شود: خروجی‌های منتظره و الگوریتم و سـازوکار تولید آن‌ها و روش مقایسـه خروجی واقعی و خروجی منتظره. فرآینـد یافتن خروجی‌های منتظره صحیح و قابل اعتماد را مسئلهٔ پیش‌گو می‌نامند.

در مقایسـه با بسـیاری از جنبه‌های خودکارسـازی آزمون، مسئله خودکارسـازی پیش‌گوی آزمون کمتر مـورد توجه قرار گرفته و یک علت آن البته به سـختی ذاتی خود مسئله برمی‌گردد. خودکارسازی تولیـد پیش‌گوی آزمون باعث جهش مهمّی در سـایر فنون خودکار تضمیـن کیفیت همچون آزمون‌های مختلف، مکان‌یابی اشـکال‌ها و تعمیر خطاها می‌گردد.

یکی از عوامل تأثیرگذار بر دشـواری مسـئله پیش‌گوی آزمون، تنوع و پیچیدگی سیستم‌ها و بن‌سـازه‌های نرم‌افزاری است. در حقیقت نرم‌افزارهای کاربـردی امروز با زبان‌های برنامه‌سـازی متعدد تولید می‌شـوند و تولید پیش‌گوی آزمون برای هر زبان مسایل مخصوصی دارد. همچنین بسیاری از نرم‌افزارها با سرهم کردن و باز به کارگیری کدهای آماده سـاخته می‌شـوند. اغلـب اوقات این کدهـا در قالب واحدهای ترجمه شـده در اختیار ما هسـتند و کـد منبع آن‌ها در دسـترس نیسـت. با توجه به این‌که نشان داده شـده فن واقع‌بینانه برای تولید پیش‌گو نیاز به تحلیل کد منبع دارد، برای چنین کدهایی تولید پیش‌گو دشـوار می‌شود و فنون مخصوصی را می‌طلبد. مشکل دیگر موقعی بروز می‌کند که خروجی منتظره مقدار مشخصی نیست و دامنه‌ای از مقادیر همگی مجاز هستند. خیلی از اوقات خروجی‌های نرم‌افزار رشته‌های نویسه‌ای هستند با حالت‌های بسیار یا اصلا متنی نیسـتند و نمایش گرافیکی دارند. برای نرم‌افزارهای شبیه‌ساز اساساً خروجی قطعی وجود ندارد و برای ورودی مشـخص چندین خروجی

معتبرند. تولید خودکار پیش‌گوی آزمون برای این دسته از کاربردها فوق‌العاده مشکل می‌شود.

راه‌حل‌ها و ارزیابی

برای تولید خودکار پیش‌گوی آزمون طبیعی است که نیاز به تحلیل کد منبع داشته باشیم. همچنین فن خودکار باید سعی کند به‌طریقی رفتار نرم‌افزار را یاد بگیرد و بداند نرم‌افزار چه می‌کند تا بتواند پیش‌گوی آزمون را تولید کند. برای تحقیق این نیازمندی، چند سال اخیر اغلب فنون پیشنهادی از روش‌های یادگیری ماشینی استفاده کرده‌اند که شبکه عصبی تاکنون از مؤثرترین آن‌ها بوده است. شبکه عصبی یا هر روش یادگیری ماشینی تعدادی ورودی آزمون و خروجی منتظره را دریافت می‌کند تا با آن‌ها رفتار برنامه را یاد بگیرد و مدلی ساخته شود. سپس با این مدل سعی می‌کنند برای سایر ورودی‌ها پیش‌گوی آزمون را پیش‌بینی و تولید نمایند. به‌همین دلیل اوزان سنجشی همچون دقت ، صحت و بازیابی و نرخ مثبت‌های کاذب و واقعی مطرح می‌شوند و پیشرفت در این حوزه با افزایش دادن دقت و صحتِ پیش‌بینی محقق می‌گردد. چون تولید پیش‌گوی آزمون برای هر نگارش برنامه فقط یک‌بار انجام می‌شود و حالت پیش پردازش دارد، بنابراین می‌توان از زمان و پیچیدگی محاسباتی فنون و الگوریتم‌های طراحی شده صرف‌نظر کرد و معیار مقایسه و برتری فنون را اوزان سنجش مذکور قرار داد.

با توجه به مشکلات عمده‌ای که در حل مسئله پیش‌گوی آزمون هست، امروزه هر فن مناسب در ابتدا فرض‌های محدود کننده بسیاری باید در نظر بگیرد تا مسئله آن‌قدر کوچک شود که بتوان آن‌را مهار و حل کرد. لذا پیدا کردن فن عمومی برای حل مسئله پیش‌گو حداقل تا چند سال آینده دور از دسترس است و فنون طراحی شده قطعاً خاص منظوره‌اند.

کاربرد

یکی از حوزه‌هایی که تولید خودکار پیش‌گوی آزمون برای آن‌ها سودمند است، نرم‌افزارهای موروثی هستند که عمدتاً به زبان C++ نوشته شده‌اند. برای نمونه شرکت گوگل حجم بسیاری از این نرم‌افزارها دارد که با هزینه بسیار تولید شده‌اند و هنوز کار می‌کنند. متأسفانه چنین برنامه‌هایی هنوز خطادارند و کاربران مایلند که هنوز از این نرم‌افزارها استفاده کنند ولی تمایل دارند خطاهای آن‌ها هم رفع شود. اشکال‌زدایی چنین نرم‌افزارهایی هزینه بسیار دارد و ساده هم نیست چون اغلب ممکن است مستندات و توصیفاتی وجود نداشته باشد، در دسترس نباشد، مبهم و ناکامل باشد یا نویسنده نرم‌افزارها در دسترس نباشند. اینجاست که فنون خودکار تولید پیش‌گوی آزمون ارزشمند و کاربردی می‌شوند.

تمام بخش‌های فرایند آزمون و اشکال‌زدایی نرم‌افزار به پیش‌گوی آزمون نیاز دارند. پس آزمون و اشکال‌زدایی کاملاً خودکار، زمانی واقعاً محقق می‌شود که این نیازمندی تأمین شود. تعمیر خطاهای نرم‌افزار آخرین و مهم‌ترین بخش اشکال‌زدایی است که خودکارسازی آن اساساً وابسته به تولید خودکار پیش‌گوست.

مراجع

[1] A. Orso and G. Rothermel, “Software Testing: A Research Travelogue (2000 – 2014),” In Proceedings of the on Future of Software Engineering, pp. 117–132. ACM, 2014.

[2] S. Anand, E. K. Burke, T. Y. Chen, J. Clark, M. B. Cohen, W. GriesKamp, M. Harman, M. J. Harrold, and P. McMinn, “An Orchestrated Survey of Methodologies for Automated Software Test Case Generation,” J. Syst. Softw., vol. 86, no. 8, pp. 1978–2001, 2013.

[3] H. Kaur and G. Gupta, “Comparative Study of Automated Testing Tools: Selenium, Quick Test Professional and TestComplete,” J. Eng. Res. Appl., vol. 3, no. 5, pp. 1739–1743, 2013.

[4] S. R. Shahamiri, W. M. N. W. Kadir, and S. Ibrahim, “A Single-network ANN-based Oracle to Verify Logical Software Modules,” ICSTE 2010 - 2010 2nd Int. Conf. Softw. Technol. Eng. Proc., vol. 2, pp. 272–276. IEEE, 2010.

[5] E. T. Barr, M. Harman, P. McMinn, M. Shahbaz, and S. Yoo, “The Oracle Problem in Software Testing: A Survey,” IEEE Trans. Softw. Eng., vol. 41, no. 5, pp. 507–525, 2015.

[6] M. Harman, P. McMinn, M. Shahbaz, and S. Yoo, “A Comprehensive Survey of Trends in Oracles for Software Testing,”Univ. Sheffield, Dep. Comput. Sci. Tech. Rep. CS-13-01, pp. 1–32, 2013.

[7] Vineeta, A. Singhal, and A. Bansal, “Generation of Test Oracles Using Neural Network and Decision Tree Model,” 2014 5th Int. Conf. - Conflu. Next Gener. Inf. Technol. Summit, pp. 313–318, 2014.

[8] M. E. Yousif, S. R. Shahamiri, and M. B. Mustafa, “Test Oracles Based on Artificial Neural Networks and Info Fuzzy Networks: A Comparative Study,” Proc. 2015 10th IEEE Conf. Ind. Electron. Appl. ICIEA 2015, pp. 467–471, 2015.

[9] S. R. Shahamiri, W. M. N. W. Kadir, S. Ibrahim, and S. Z. M. Hashim, “An Automated Framework for Software Test Oracle,” Inf. Softw. Technol., vol. 53, no. 7, pp. 774–788, 2011.

[10] P. A. Nardi and E. F. Damasceno, “Survey on Test Oracles,”J. Adv. Theor. Appl. Informatics (ISSN 2447-5033), vol. 1, no. 1, pp. 50–59, 2015.

[11] B. P. Lamancha, M. Polo, D. Caivano, M. Piattini, and G. Visaggio, “Automated Generation of Test Oracles Using a Model-driven Approach,” Inf. Softw. Technol., vol. 55, no. 2, pp. 301–319, 2013.

مسابقه عملی دانشجویی

در دانشکده ریاضی، آمار و علوم کامپیوتر پردیس علوم دانشگاه تهران

مسابقه طراحی و ساخت دستگاه تشخیص حالت چهره روز سه شنبه ۵ بهمن ماه با حضور ۶ تیم از دانشجویان دوره کارشناسی ارشد رشته علوم کامپیوتر دانشکده ریاضی، آمار و علوم کامپیوتر دانشگاه تهران به عنوان یکی از تمرین‌های عملی درس پردازش تصویر به راهنمایی خانم دکتر هدیه ساجدی در پردیس علوم دانشگاه تهران برگزار شد. این رقابت در قالب حمایت از مسابقات عملی برنامه جوانه پارک علم و فناوری دانشگاه تهران و با حضور نماینده پارک برگزار شد و در پایان به ۳ تیم برتر مسابقه جوایزی از طرف پارک علم و فناوری دانشگاه تهران اعطا شد. همچنین هزینه‌های ساخت دستگاه توسط گروه‌های دانشجویی تا سقف مجاز مطابق با دستورالعمل مربوطه از محل اعتبار مسابقات عملی دانشجویی پارک علم و فناوری دانشگاه تهران، تامین و به تیم‌های دانشجویی پرداخت شد.

پارک علم و فناوری دانشگاه تهران با هدف ترویج تفکر خلاق و آشنایی دانشجویان دانشگاه تهران با انجام کامل یک پروژه تجربی دارای رویکرد نیاز بازار، از برگزاری مسابقات دانشجویان بر پایهٔ دروس عملی حمایت می‌کند. اعضای هیئت علمی دانشگاه می‌توانند در دروسی که دارای کار عملی هستند، ضمن رعایت دستورالعمل مربوطه در پارک علم و فناوری، ساخت یک محصول مشخص را در میان دانشجویان به مسابقه بگذارند. حمایت مادی پارک علم و فناوری از هزینه ساخت محصول تیم‌های حاضر در مسابقه و اهدای جوایز نقدی به تیم‌های اول تا سوم از جمله خدمات پارک علم و فناوری در این مسابقات می‌باشد. فراخوان مسابقات عملی دانشجویی در ابتدای هر نیمسال تحصیلی به تمامی اعضای هیئت علمی دانشگاه تهران ابلاغ می‌شود.

مسابقه طراحی و ساخت دستگاه تشخیص حالت چهره

سومین مسابقه عملی دانشجویی، با درخواست خانم دکتر هدیه ساجدی عضو هیئت علمی پردیس علوم برگزار شد. ۶ تیم از دانشجویان دوره کارشناسی ارشد رشته علوم کامپیوتر در مسابقه‌ای با عنوان «طراحی و ساخت دستگاه تشخیص حالت چهره» به‌عنوان یکی از تمرین‌های عملی درس «پردازش تصویر» در تاریخ ۹۵/۱۱/۰۵ در محل پردیس علوم شرکت کردند. مسابقه با حضور هیئت داوران، مدرس واحد درسی و نماینده پارک علم و فناوری برگزار و در پایان از ۳ تیم برتر تقدیر شد. همچنین تا سقف ۳،۰۰۰،۰۰۰ ریال از هزینه‌های هر تیم برای ساخت دستگاه از طرف پارک علم و فناوری بر اساس اسناد و مدارک مالی پوشش داده شد.

شرح مبسوط این مسابقه در ادامه گزارش آمده است.

۱. مشخصات مسابقه

نام و نام خانوادگی مدرس : هدیه ساجدی	مرتبه علمی: استادیار
نام دانشکده : ریاضی، آمار و علوم کامپیوتر	
عنوان درس: پردازش تصویر	
مقطع تحصیلی دانشجویان: کارشناسی ارشد	
عنوان مسابقه: طراحی و ساخت دستگاه تشخیص حالت چهره	
تعداد تیم‌ها : ۶	تعداد اعضای هر تیم: ۱ الی ۲ نفر
زمان مسابقه: ۹۵/۱۱/۰۵	مکان مسابقه: پردیس علوم

۲. اعضای کمیته داوری

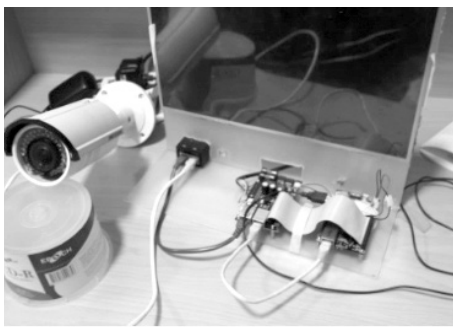
دکتر هدیه ساجدی، عضو هیئت علمی دانشکدهٔ ریاضی، آمار و علوم کامپیوتر

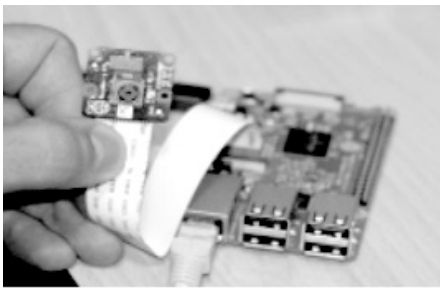
دکتر سید مجتبی مجتهدی، عضو هیئت علمی دانشکدهٔ ریاضی، آمار و علوم کامپیوتر

۳. برگزیدگان

شرح	شماره تیم	نام سرپرست تیم	جایزه
رتبه اول	۵	مریم بهرامی	۵,۰۰۰,۰۰۰ ریال نقدی
رتبه دوم	۶	محمد آزاده	۳,۰۰۰,۰۰۰ ریال نقدی
رتبه سوم	۱	مهدی عزیزی	۲,۰۰۰,۰۰۰ ریال نقدی

۴. مشخصات تیم‌ها

شماره تیم	اسامی اعضا	سرپرست تیم	تصویر محصول
۱	مهدی عزیزی	مهدی عزیزی	
۲	محمد صالح وحدت پور	محمد صالح وحدت پور	
۳	علیرضا فریدونی	علیرضا فریدونی	

۴	محمد حسین احمدی مجد سید امیرحسین رحیمی	محمد حسین احمدی مجد	
۵	مریم بهرامی	مریم بهرامی	
۶	محمد آزاده فرزاد رمضانی	محمد آزاده	

۵-مشخصات فنی سیستم برنده مسابقه

سیستم تشخیص احساس چهره، شامل یک دوربین MIMI HD می‌باشد که تصاویر را به صورت آنالوگ دریافت می‌کند. در این سیستم از یک مبدل خارجی برای تبدیل تصاویر آنالوگ به دیجیتال استفاده می‌شود. تصاویر دیجیتال به صورت برخط، ورودی الگوریتم طراحی شده می‌باشند. در این الگوریتم که به زبان برنامه‌نویسی متلب پیاده‌سازی شده است، ابتدا چهره اشخاص موجود در تصویر تشخیص داده می‌شود. سپس قسمت لب شناسایی می‌گردد. در ادامه، از قسمت لب، برداری استخراج می‌شود که در تشخیص احساس چهره به کار گرفته می‌شود. برای این منظور، پایگاه داده‌ای متشکل از قسمت لب در حالت‌های خوشحال، ناراحت و خنثی تهیه شده است و از آن برای آموزش یک شبکه عصبی دو لایه استفاده می‌شود. در نهایت، برای آزمایش سیستم از یک تصویر گرفته شده توسط دوربین، بردار استخراج شده از قسمت لب به شبکه‌عصبی داده می‌شود و به این صورت حالت چهره شناسایی می‌گردد.

خروجی این سیستم، بیان احساس چهره در سه حالت خوشحالی، ناراحتی و خنثی است.

تصاویر مسابقه

پیدایش مهندسی نرم‌افزار

نیکلاوس ویرث و مهندسی نرم‌افزار

(بخش هفتم)

نوشتهٔ ادگار دِی لایت

ترجمهٔ: ابراهیم نقیب‌زادهٔ مشایخ

پست الکترونیکی: mashayekh@isi.org.ir

پیش‌گفتار مترجم

در مجلـهٔ CACM مربوط بـه ژانویهٔ ۲۰۱۵، مقاله‌ای توجهـم را جلب کرد با عنوان «اشک‌های دونالد کنوث». این مقاله که توسط توماس هِی نوشته شده بود، درواقع انتقادی بود به مطالبی که کنوث در سـخنرانی خود در دانشگاه استنفورد بیان کرده بود. در آن سـخنرانی، کنوث به شدّت به مقالهٔ مارتین کمپبل- کلی با عنوان «تاریخِ تاریخ نرم‌افزار» تاخته بود و از این که او در مقاله‌اش سـطح تاریخ رایانش و علوم کامپیوتر را این قدر پایین آورده به شدّت انتقاد کرده بود. کنوث در پایان سـخنرانیش گفته بود که آنقدر از خوانـدن آن مقاله ناراحت و غمگین شده بود که شیشه‌های عینکش از اشک‌های چشمش لک شده بود و او به سختی توانسته خواندن مقاله را به پایان برد.

تومـاس هِی در مقاله‌اش ضمن تمجید از کنوث و دسـتاوردهای بی‌مانندش در حوزهٔ رایانش، از این که کتاب‌های کمی دربارهٔ تاریخ علوم کامپیوتر نوشته شده با او همداسـتانی کرده امّا این کوتاهی را بیشتر متوجه خود دانشمندان این رشته دانسته است. او گفته: «امروز شـما می‌توانید در دانشکده‌های ریاضی و پزشکی، حقوق و چند رشـتهٔ دیگر مدرک دکتری بگیرید و موضوع رسـاله‌تان، تاریخچهٔ آن رشته باشـد. امّا چنین امکانی در دانشـکده‌های علوم کامپیوتر وجود ندارد. من دانشـکدهٔ علوم کامپیوتری را در آمریکا نمی‌شناسـم که کسی را که موضوع رسـالهٔ دکتریش، جنبه‌های تاریخی علوم کامپیوتر بوده، اسـتخدام کرده باشد. همچنین کسـی را در آمریکا نمی‌شناسـم که به خاطر کارهایش در زمینهٔ تاریخ علوم کامپیوتر در یک دانشکدهٔ علوم کامپیوتر تشویق شده یا ارتقاء یافته باشد.»

مقالهٔ ۵ صفحه‌ای توماس هِی بسیار خواندنی است و علاقه‌مندان به این موضوع را بـه خواندن متـن کامل آن توصیه می‌کنـم. او دارای درجهٔ کارشناسـی علوم کامپیوتـر از دانشـگاه منچسـتر (۱۹۹۵) و دکتری تاریخ و جامعه‌شناسـی علم از دانشـگاه پنسـیلوانیا (۲۰۰۳) اسـت. امّا غرض از آوردن این مقدمه این بود که در مقالهٔ مزبور به چند کار ارزشـمند نیز که در این زمینه انجام شـده اشـاره گشته که از آن جمله، کتاب «پیدایش مهندسی نرم‌افزار: از تورینگ تا دایکسترا» نوشتهٔ ادگار دِی لایت است. عنوان این کتاب برای من که نزدیک ۳۰ سال است درس مهندسـی نرم‌افزار را در دانشـگاه تدریس می‌کنم، جالب بود. جستجویی در اینترنت کردم و دیدم که افراد سرشناسـی چون گریدی بوچ و جان رینولدز نقدهای بسـیار خوبی دربارهٔ این کتاب نوشته‌اند. نویسندهٔ کتاب، ادگار دِی‌لایت نیز خود دارای درجهٔ کارشناسـی ارشد منطق از دانشـگاه آمستردام (۲۰۰۹) و دکتری مهندسـی نرم‌افزار از دانشـگاه لوون بلژیـک (۲۰۰۶) و در حال حاضر پژوهشگر تاریخ رایانش در دورهٔ پسادکتری در دانشگاه فناوری آیندهوون هلند است. کتاب را توسط دوسـتی در کانادا از کتاب‌فروشی آمازون خریداری کردم و ظرف یکی دو هفته توسـط مسافری به دستم رسید. خودم از خواندن آن لذت بردم و آن را به دو تن از همکارانم در دانشـکدهٔ علوم کامپیوتر دانشـگاه تهران نیز نشـان دادم و آن‌ها هم مطالب مطرح شـده در کتاب را برای پژوهشگران و متخصصان علوم کامپیوتر و به ویژه دانشجویان این رشته بسیار مفید دانستند. از این رو به ترجمهٔ کتاب اقدام کردم که به صورت یک سلسـله مقالهٔ دنباله‌دار در این نشریه به نظرتان خواهد رسید.

* * *

من برای درک بهتر تاریخچهٔ زبان‌های برنامه‌نویسی، گفتگویی را با نیکلاوس **ویرث**[1]، برندهٔ جایزهٔ تورینگ در سال ۱۹۸۴ ترتیب دادم. **ویرث** در رشتهٔ مهندسی برق تحصیل کرد و بعداً به طراح موفق زبان‌های برنامه‌نویسی تبدیل شد. او خالق زبان‌های برنامه‌نویسی اویلر، الگول W، پاسکال، ماجولا و اُبرون است.

متن گفتگوی ما که در این فصل ارائه شده است نشان می‌دهد که **ویرث** در خلال دههٔ ۱۹۶۰ به خوبی از دو دستگی بین آراستگی و زیبائی ریاضی از یک سو و کارایی ماشین از سوی دیگر، آگاه بوده است. او برای ارتباط برقرار کردن و آموزش دادن مؤثر، اوّلی را برگزید امّا همچنین تأکید کرد که در اصل و بیش از همه او یک مهندس است و خواهان کارکردن واقعی ماشین‌ها می‌باشد. برای نمونه، **ویرث** به یاد می‌آورد که به خاطر دستیابی به راه‌حلّی ساده، استفاده از یک پشتهٔ زمان اجرا را با وجودی که به قیمت از دست رفتن کارایی ماشین می‌شد، ترجیح داده بود. همچنین درخواست او برای زبان مقصد[2] مستقل از ماشین به خاطر حمل‌پذیری[3] برنامه‌های پاسکال، بسیار شبیه کارهای دایکسترا در زمینهٔ الگول ۶۰ است.

در خلال اوایل دههٔ ۱۹۷۰، **ویرث** به مقالهٔ ۱۹۶۹ هور[4] با عنوان «مبنای اصل موضوعی[5] برای برنامه‌نویسی رایانه»[۱۲۵] علاقه‌مند شد و با او در تلاش برای به کاربستن معناشناسی[6] بر پایهٔ اصل موضوع بر روی زبان برنامه‌نویسی پاسکال، همکاری کرد. در سال‌های بعد، در پی موفقیت‌های اولیه‌اش، اذعان کرد که چنین تلاش‌هایی برای صورت‌بندی[7]، مزایای عملی محدودی به همراه خواهد داشت و این که «اغلب برنامه‌نویسان با ریاضیات محض آشنایی چندانی ندارند.» او در نوشته‌های بعدیش به وضوح نسبت به موج پر نوسان صورت‌بندی در رایانش دههٔ ۱۹۶۰ و پس از آن، واکنش نشان داد. او امروز نیز عقیده دارد که شکاف بین نظریه و عمل در حال گسترش است. به گفتهٔ خود او:

«شیوه‌ای که نظریه‌پردازان به وجود آورده‌اند (روش‌های تحلیلی برای اثبات صحّت برنامه) شبیه یک ابر نظری است که از جایی که قرار بوده باشد، حرکت کرده و به جای دیگری رفته است.» (بخش ۵-۲-۲ را ببینید)

موضوعات دیگری که با **ویرث** مورد بحث قرار داده‌ام عبارتند از بحران نرم‌افزار[8] در سال ۱۹۶۸ و نرم‌افزار نهفته[9].

ملاقات با ویرث

گفتگوی من با **ویرث** در ۹ نوامبر ۲۰۱۰ در شهر زوریخ در سوئیس، صورت گرفته است. پیاده‌سازی نوار گفتگو توسط من انجام شده و سپس توسط **ویرث** و خودم، ویرایش مختصری گردیده است.

زندگی‌نامهٔ مختصر نیکلاوس ویرث

نیکلاوس **ویرث** در سال ۱۹۳۴ در کشور سوئیس به دنیا آمده است. او درجهٔ کارشناسی ارشد مهندسی برق را در سال ۱۹۵۸ از دانشگاه فنی زوریخ (ETH) دریافت کرد و در پی آن، برای مدّت کوتاهی به‌عنوان دستیار آموزشی[10] در دانشگاه لاوال در ایالت کِبِک در کشور کانادا به کار پرداخت و سپس به دانشگاه برکلی در کالیفرنیا رفت تا دورهٔ دکتری زبان‌های برنامه‌نویسی را زیر نظر استاد هَری هاسکی بگذراند (۱۹۶۰-۱۹۶۳). **ویرث** از سال ۱۹۶۳ تا ۱۹۶۷ به‌عنوان استادیار در دانشگاه استنفورد، در دانشکدهٔ تازه تأسیس دانش‌رایانه (علوم کامپیوتر) به فعالیت آموزشی مشغول شد و در سال ۱۹۶۷ به زوریخ بازگشت و کوته‌زمانی پس از آن، به همراه چند تن دیگر، دانشکدهٔ دانش رایانه (علوم کامپیوتر) را در دانشگاه فنّی زوریخ بنیان گذاشت. او دو نوبت فرصت مطالعاتی خود را (۱۹۶۷ تا ۱۹۶۸ و ۱۹۸۴ تا ۱۹۸۵) در شرکت زیراکس گذراند و در ماه مارچ ۱۹۹۹ با درجهٔ استادتمام از دانشگاه فنّی زوریخ بازنشسته شد. **ویرث** جوایز گوناگونی دریافت کرده که از آن جمله می‌توان به جایزهٔ امانوئل پیور انجمن مهندسان برق و الکترونیک (IEEE) در سال ۱۹۸۳، جایزهٔ تورینگ در سال ۱۹۸۴ و جایزهٔ پیشتازان رایانهٔ IEEE در سال ۱۹۸۸ اشاره کرد.

۵-۱ مهندسی: از سرگرمی تا حرفه

دِی‌لایت: شما در سال ۱۹۳۴ در شهر وینترهور سوئیس به دنیا آمدید. آیا این در بخش آلمانی زبان کشور سوئیس است؟

ویرث: بله، در ۳۰ کیلومتری شمال زوریخ.

دِی‌لایت: آیا هنوز هم همانجا زندگی می‌کنید؟

ویرث: نه. امّا ۲۵ سال نخست عمرم را آنجا گذراندم. در خلال دوران تحصیلم در دانشگاه فنّی زوریخ، هر روز ۲۰ دقیقه با قطار از وینترهور به زوریخ می‌رفتم و برمی‌گشتم.

دِی‌لایت: آیا جنگ جهانی دوم را به یاد می‌آورید؟ کشور سوئیس درگیر جنگ نبود، مگر نه؟

ویرث: رسماً نه، ولی تحت تأثیر جنگ بود.

دِی‌لایت: چه تأثیری بر روی زندگی شما داشت؟

ویرث: خوب، البته من یک پسر جوان بودم. امّا می‌توانستم نگرانی و تنش را در فضای جامعه حس کنم. پدرم بیشتر وقت‌ها در گارد مرزی خدمت می‌کرد و من و مادرم تنها زندگی می‌کردیم. مواد غذایی سهمیه‌بندی شده بود و نسبتاً کمیاب بود. ما در حیاط خانهٔ خودمان سبزیجات می‌کاشتیم. و البته صدای هواپیماهای جنگی را

1- Niklaus Wirth
2- object language
3- portability
4- Hoare
5- axiomatic
6- semantics
7- formalization
8- software crisis
9- embedded software
10- teaching assistant

که شـــب و روز از روی سرمان رد می‌شدند می‌شنیدیم. شرایط خیلی بد و ناراحت‌کننده‌ای بود.

۵-۱-۱ شور و شوق اوّلیه برای مهندسی

دِی‌لایت: در پایان جنگ، شما یازده ساله بودید. اوضاع و احوال شما در ســال‌های نخســتین پس از جنگ چگونه بود؟ آیا نسبت به چیز دیگری، مثلاً فوتبال، شــور و شوق داشــتید که بعداً این اشتیاق به تدریج به سوی مهندسی کشیده شد؟

ویــــرث: نه. وضعیت من خیلی روشـــن بود. از همـــان اوان کودکی، همیشه به ماشـــین‌آلات و اسباب‌بازی‌ها علاقه‌مند بودم. کمی بعد به خواندن کتاب‌های فیزیک و شـــیمی که در کتابخانهٔ پدرم پیدا کرده بودم، پرداختم. علاقه‌مندی‌های من تماماً به علوم فنی بود.

دِی‌لایت: شما خواهر و برادری هم داشتید؟

ویرث: نه، تنها بودم. باید برای ســـرگرم نگاه داشتن خودم، چیزهای جدید یاد می‌گرفتم.

دِی‌لایت: آن کتاب‌ها را چگونـــه می‌خواندید؟ آیا فقط به طور گذرا آن‌ها را می‌خواندید؟

ویرث: من همیشه عاشق آزمایش کردن بودم. در دوران نوجوانی، به قطار خیلی علاقه داشتم. از کنار خانهٔ ما چهار خط قطار رد می‌شد. من معمـــولاً به آن‌ها نگاه می‌کردم و در ذهنم به برنامه‌ریزی می‌پرداختم. آرزو می‌کردم که یک خط آهن مدل داشته باشم، که البته نداشتم و بعداً به مدل هواپیما علاقه‌مند شـــدم. در سن ۱۲ سالگی، این موضوع را فعّالانـــه دنبال می‌کـــردم. ابتدا هواپیماهای مدل را از روی نقشـــه می‌ساختم و جالب‌تر آن که شروع کردم به طراحی‌کردن هواپیماهای خودم. در سن ۱۵ سالگی در تیم ملّی قرار گرفتم و برای یک مسابقهٔ بین‌المللی به انگلســـتان فرستاده شـــدم. دوران پرمشغله‌ای برای من بـــود. اکنون که به گذشـــته نگاه می‌کنم، تعجّـــب می‌کنم که چگونه دوران دبیرستان را بدون هیچ مشکلی به پایان رساندم.

پـــس از گذراندن درس شـــیمی در مدرســـه، در زیرزمین خانه‌مان یک آزمایشگاه شیمی درست کردم. این دوران علاقه‌مندی به شیمی بود. سپس دوران سیستم‌های کنترل از راه دور برای هواپیماها فرا رسید. من یکی از نخستین کسانی در سوئیس بودم که یکی از آن‌ها را از انگلستان سفارش دادم. البته تمام تلاش‌هایم با شکست روبرو می‌شد و بدین ترتیب بود که مجبور شدم به مطالعهٔ الکترونیک بپردازم. از یکی از دوستانم کتابی دربارهٔ الکترونیک گرفتم. سرانجام، زمانی فرا رسید که باید تصمیم می‌گرفتم در چه رشته‌ای درس بخوانم. به‌طور طبیعی، الکترونیک را انتخاب کردم زیرا چیـــزی بود که همان موقع به آن نیاز داشـــتم. بدین ترتیب بود که دورهٔ چهارسالهٔ مهندسی برق را در دانشگاه فنّی زوریخ گذراندم.

دِی‌لایت: سال ۱۹۵۸ بود؟

ویرث: بله، درست است.

۵-۱-۲ مهاجرت به آمریکای شمالی

دِی‌لایت: شما پس از تحصیل در دانشگاه فنّی زوریخ به کِبِک رفتید.

ویرث: بلـــه. پدرم به من توصیه کرد که پـــس از خاتمهٔ تحصیلم به خارج از کشـــور بروم. او توصیه کرد برای حداقل یک یا دو ســـال از کشـــور خارج شـــوم و یک جای دیگر دنیا را ببینم. یادآوری می‌کنم که در خلال جنگ، مرزها بسته بود و پس از جنگ، کم‌کم داشت باز می‌شد. من نصیحت او را پذیرفتم. او مرا متقاعد کرد که باید بلافاصله بعـــد از خاتمهٔ تحصیلاتم در دانشـــگاه ایـــن کار را بکنم. زیرا در غیر این‌صورت، احتمالاً کاری در ســـوئیس خواهم گرفت، خانواده تشکیل خواهم داد و دیگر مسافرت برایم سخت خواهد شد. من هم همان‌طور کـــه او گفته بود عمل کردم و با نیّـــتِ یافتن کاری به‌عنوان مهندس، بـــه کانادا مهاجرت کردم. در ابتـــدای کار، وضعیت نومیدکننده بود و من نتوانســـتم چنین کاری پیدا کنم امّا سرانجام توانستم در دانشگاه لاوال به‌عنوان دستیار آموزشی استخدام شوم. بدین ترتیب بود که من سرانجام، کم و بیش به‌طور اتفاقی، ساکن کِبِک شدم. به خودم گفتم اگر از این کار خوشم نیامد، از همانجا به دنبال کار دیگری می‌گردم.

دِی‌لایت: چطور شد که پس از دانشگاه لاوال، به هاسکی در کالیفرنیا رسیدید؟

ویرث: خوب، من از ابتدا برای کار مهندســـی برنامه‌ریزی کرده بودم، امّا در خلال آن سال در دانشگاه لاوال، از سوی دانشجویان تحصیلات تکمیلی تشویق شدم که به تحصیلاتم ادامه دهم و وارد دورهٔ دکتری شـــوم. آن‌ها به من گفتند که درجهٔ کارشناسی ارشد نباید پایان کار تحصیلی من باشـــد. ضمناً تعداد دانشجویان تحصیلات تکمیلی فقط شـــش نفر بود: یک چینی، یک پاکستانی، یک برزیلی، یک فرانسوی، یک کانادایی و من. در ابتدا من موافق ایدهٔ دکتری نبودم و تحصیلاتی کـــه کرده بودم برایم بـــس بود. امّا بعد، پس از کمـــی فکر کردن، بر آن شـــدم تا برای گرفتن پذیرش اقدام کنم. فکر می‌کنم درخواستم برای دورهٔ دکتری را برای ۵ دانشـــگاه در آمریکا فرستادم. درخواست پذیرشم همراه با درخواست دستیاری بود زیرا باید درآمد می‌داشتم. در چهار دانشـــگاه پذیرفته شـــدم که یکی از آن‌ها دانشگاه برکلی در کالیفرنیا بود. در آنجا به من دستیاری استاد نظریهٔ اطلاعات هم داده شـــده بود. البته این چیزی که خیلی مورد علاقهٔ من باشد نبود چون ریاضیات محض بود. بنابراین در خلال همان نیمســـال اوّل به دنبال موقعیت دســـتیاری دیگری گشـــتم. بعد فهمیدم که چنین موقعیتی در زمینهٔ رایانش زیر نظر اســـتاد هاسکی در حال به‌وجود آمدن است و رایانش نیز یکـــی از موضوعاتی بود که به آن علاقه‌مند بودم. بدین ترتیب بود که به هاسکی رسیدم.

دِی‌لایت: هاسکی را چگونه توصیف می‌کنید؟ تا آن زمان، او سفرهای زیادی کرده بود و مثلاً برای آلن تورینگ هم کار کرده بود. آیا او آدم بینش‌وری بود؟ آیا تأثیر زیادی بر روی شما گذاشت؟

ویرث: او مرد آرامی بود. شـــهرتش فکـــر می‌کنم عمدتاً به خاطر کار بر روی نخســـتین رایانهٔ ساخته شده در لس‌آنجلس بود. فکر می‌کنم نام آن رایانه Whirlwind (گردباد) بود. او بعداً در شـــرکت بندیکس بر روی طراحی ماشین G15 کار کرده بود (که نخستین ماشینی بود که

من بر روی آن برنامه‌نویسی کردم). البته سرش هم خیلی شلوغ بود. او رئیس انجمن ماشین‌های رایانشی (ACM) و چند سازمان دیگر بود. بنابراین اگر راستش را بخواهم بگویم، نه، تأثیر زیادی بر روی من نگذاشت. کسی که نقش او را بر عهده گرفت، وَن‌واین‌گاردن بود که برای گذراندن فرصت مطالعاتی برای یک ترم به برکلی آمده بود. چگونگی ارتباط پیدا کردن من با او بدین ترتیب بود که یک گروه سه نفره از دانشجویان دکتری را پیدا کردم که در گوشهٔ کوچکی از طبقات پایین دانشکدهٔ ریاضی کار می‌کردند- من باید به آنجا می‌رفتم زیرا مرکز رایانه در آنجا بود. کاری که آن‌ها می‌کردند، خیلی عجیب به نظرم آمد امّا آن را جالب یافتم. آن‌ها برنامه‌نویسی می‌کردند- و من در حال آموختن برنامه‌نویسی به زبان فرترن بودم- و برنامه‌ای نوشته بودند که می‌توانست یک زبان برنامه‌نویسی را پردازش کند، زبان خودشان. یک برنامهٔ مترجم (کامپایلر) بود، حالا ما می‌دانیم که آن را چه بنامیم، مترجمی برای NELIAC که گونهٔ اوّلیهٔ الگول ۵۸ بود. این استاد هاسکی بود که آن زبان برنامه‌نویسی را از یک پایگاه پژوهشی نیروی دریایی به آن‌ها معرفی کرده بود. بدین ترتیب بود که من درگیر برنامه‌نویسی و نرم‌افزار شدم. در واقع، من به برکلی آمده بودم که زیر نظر هاسکی، سخت‌افزار و چگونگی ساخت رایانه‌ها و چگونگی کارکردن با آن‌ها را بیاموزم. امّا دیگر زمان مناسبی برای ساخت رایانه‌ها در دانشگاه‌ها نبود. آن‌ها تازه ساخت یک رایانهٔ عظیم را به پایان رسانده بودند. بنابراین، من به سوی نرم‌افزار سوق داده شدم.

۵-۱-۳ برنامه‌های مترجم

دِی‌لایت: اجازه دهید دربارهٔ راه‌های مختلف تعریف معنای[11] یک زبان برنامه‌نویسی صحبت کنیم. وَن‌واین‌گاردن از رویکرد بسیار پویایی استفاده می‌کرد که نهایتاً به الگول۶۸ انجامید. امّا در آن زمان، هنگامی که هر دو در برکلی بودید، آیا آن نگاه کاملاً مستقل از ماشین به زبان برنامه‌نویسی را به دست آورده بود؟ آیا این آن چیزی بود که بر روی آن کار می‌کرد؟ یا تأثیری که از او پذیرفته‌اید مربوط به چیز دیگری است؟

ویرث: بله، بله. یادم می‌آید که در آن موقع، کارش به نام «الگول تعمیم یافته»[12] منتشر شده بود[۳۰۱].

دِی‌لایت: فکر می‌کنم ۱۹۶۲ بود.

ویرث: شاید حتی ۱۹۶۱. این هنگامی است که او به برکلی آمده بود و ما دربارهٔ این موضوعات بحث می‌کردیم. در آن موقع، من چگونگی طراحی برنامه‌های مترجم را یاد گرفته بودم. من با مشاهدهٔ این که برنامهٔ مترجم NELIAC دارای به‌هم‌ریختگی‌های بسیاری بود و تنها یک نفر بود که از آن سر در می‌آورد (که او هم کارول کان بود)، با خودم فکر کردم که نظم و ترتیب‌بخشی به فرایند طراحی برنامهٔ مترجم، می‌تواند عنوان جذّابی برای پایان‌نامهٔ دکتری من باشد. از سوی دیگر، وَن‌واین‌گاردن با نسخهٔ کوچک‌شدهٔ الگولش، الگول تعمیم یافته، مرا تحت تأثیر قرار داده بود. رویکرد او مستلزم قاعده‌های کمتر امّا عمومی‌تر بود. بدین ترتیب، این هر دو چیز در هم آمیخت. یادآوری می‌کنم که من در اصل و بیشتر از هر چیز، یک مهندس بودم: علاقه‌ای به نوشتن مقاله نداشتم، فقط می‌خواستم ماشینی کار کند. بنابراین، من نه تنها زبانی را که اویلر[13] نامیدمش و بیشتر در راستای خطوط کلی الگول تعمیم یافته بود، طراحی کردم، بلکه آن را پیاده‌سازی هم کردم. پیاده‌سازی اویلر مهم بود زیرا تنها در این صورت بود که برای من واقعاً وجود می‌داشت. این کار به رسالهٔ من در سال ۱۹۶۳ در برکلی انجامید. البته تعریف اویلر، اساساً به همان روش قدیمی ماشین‌انگارانه[14] بود، هر چند من (مانند سایر افراد در آن زمان) نگفتم که تعریف، در پیاده‌سازی آن قرار دارد. در عوض، سازوکاری را برای تعریف معنا بر حسب یک ماشین خیلی ساده‌تر که بعداً برای اجرای مفسّر[15] زبان مورد استفاده قرار گرفت، ابداع کردم. البته باید اعتراف کنم که راه‌حل ارضاءکننده‌ای نبود امّا در آن زمان چیزی بهتر از آن به ذهنم نمی‌رسید.

دِی‌لایت: منظورتان سیستم ساختار عبارت[16] [۳۱۲]است؟

ویرث: بله، من بر روی تجزیهٔ نحو[17] نیز کار کردم. در آن زمان، موضوع خیلی داغی بود و البته مبنایی برای سامانمندتر کردن برنامهٔ مترجم.

دِی‌لایت: درست است زیرا پس از آن «زبان برنامه‌نویسی ساختار عبارت» شما را داریم[۳۱۲]. لطفاً مرا تصحیح کنید اگر اشتباه می‌کنم، آیا این مستقیماً به آنچه شما رایانهٔ پشته‌ای فرضی[18] نامیده‌اید[۲۷۰] مربوط است؟

ویرث: بله، این همان ماشین ساده‌تر بود که صحبتش را کردم.

دِی‌لایت: این ایده از کجا آمد؟ آیا ابداع خودتان بود؟

ویرث: مایل نیستم این را ابداع خودم بنامم. منظورم این است که ایدهٔ پشته پیش از آن نیز مطرح بود. امّا شاید من نخستین کسی بودم که زبانی را برحسب یک زبان بسیار ساده‌تر تعریف کردم. من این ایده را بعداً برای کد پاسکال P و کد ماجولا M بازبه‌کارگیری[19] کردم.

دِی‌لایت: هاسکی هم در سال ۱۹۶۲ مقاله‌ای دربارهٔ زبان‌های میانی[20] مستقل از ماشین نوشته است[۱۴۲,۱۵۲]. فکر می‌کنم او هم در زمینهٔ برنامه‌های مترجم خبره بود.

ویرث: نه، او واقعاً مرد سخت‌افزار بود.

دِی‌لایت: پس این‌طور. آیا وَن‌واین گاردن هنگامی که با او در برکلی ملاقات کردید، با شما دربارهٔ کارهای دایکسترا در زمینهٔ برنامهٔ مترجم الگول ۶۰ صحبت کرد؟

ویرث: (با مکث) نه. (باز هم مکث) من چه موقع دایکسترا را شناختم؟

11- semantic
12- Generalized ALGOL
13- EULER
14- mechanistic
15- interpreter
16- Phrase Structure System
17- syntax parsing
18- Hypothetical Stack Computer
19- reuse
20- intermediate

من برای نخســـتین بار دایکسترا را در کنگرهٔ IFIP (فدراسیون جهانی پردازش اطلاعات) در مونیخ در سال ۱۹۶۲ ملاقات کردم. من دربارهٔ او از همان گروه کوچک ســـه نفره که بر روی برنامهٔ مترجم NELIAC کار می‌کردند، شـــنیده بودم. (دِی لایت مقالهٔ NELIAC [۱۴۳] را به **ویرث** نشـــان می‌دهد.) اوه، شما آن را همین جا دارید: هاسکی، لاو و **ویرث** نویســـندگان این مقاله هستند، امّا شخصی که در اصل درگیر برنامـــهٔ مترجم بود، کارول کان بود. و همین‌طور بیل کیز. این دو نفر، سمیناری را برگزار کردند و یک چیز خیلی تأثیرگذار در آن سمینار، گزارش الگول ۶۰ بود [۱۰]. این مربوط می‌شد به اوایل ۱۹۶۱. من در پائیز ۱۹۶۰ به برکلی آمده بودم. ما شـــروع به تحلیل الگول ۶۰ کرده بودیم و در خلال این تلاش، من چیزهایی دربارهٔ دایکسترا می‌شنیدم و بعـــد از آن، کنگـــرهٔ IFIP در مونیخ در ســـال ۱۹۶۲ بود. من از این فرصت استفاده کردم تا به مونیخ بروم و دایکسترا را نیز ملاقات کنم. او مقالـــه‌ای در آنجا ارائه کرد[۶۴] که بســـیار تأثیرگـــذار بود. امّا نه، خود وَن‌واین‌گاردن هنگامی که او را در برکلی ملاقات کردم، صحبت زیادی دربارهٔ کار دایکسترا در زمینهٔ برنامه‌های مترجم نکرد.

دِی‌لایت: آیا فکر می‌کنید که این شـــاید به خاطر این بوده باشد که وَن‌واین‌گاردن به اندازهٔ دایکسترا و زونِولد درگیر ساخت برنامهٔ مترجم نبود؟

ویرث: بله، من در سال ۱۹۶۵ به آمستردام رفتم و دو ماه را در مرکز ریاضیات گذراندم. دایکسترا دیگر آنجا نبود. فکر می‌کنم زونِولد بود و نِدرکورن را هم به یاد می‌آورم.

دِی‌لایت: چیزی که تعجّب مرا برمی‌انگیزد این اســـت که دایکسترا و زونِولد، با برنامهٔ مترجم الگول ۶۰ خودشـــان، یک زبان مقصد میانی با یک پشتهٔ عمومی، خیلی شـــبیه به کد P بعدی، نیز معرفی کرده‌اند. بنابراین مـــن فکر می‌کنم این تکنیک باید به‌طور همزمان در جاهای مختلفی ابداع شده باشد.

ویرث: ممکن اســـت این‌طور باشد. من فکر می‌کنم ایدهٔ واضحی بود. مـــن نمی‌خواهم ادّعای مالک‌بودن آن را بکنم. امّا یادم نمی‌آید که آن را از کس دیگری یاد گرفته باشم.

دِی‌لایت: کسان زیادی نبودند که آن اوایل این کار را انجام دادند.

ویرث: خوب، رایانه‌ها خیلی کند بودند. امروز داشتن مفسّر مسئله‌ای نیست. البته پاســـکال P کمی بعدتر آمد، امّا روشی برای داشتن یک برنامـــهٔ مقصد قابل‌حمل[۲۱] بود که به افراد کمک می‌کرد تا برنامه‌های مترجم پاسکال را به ماشین‌هایشان حمل کنند.

دِی‌لایت: آیا یادتان می‌آید که در مورد پاســـکال، ماشین میانی شما یک پشته داشت یا بیشتر؟

ویرث: یکی.

دِی‌لایت: چرا این برای شـــما یک اصل اساسی بود؟ آیا داشتن تنها یک پشته الزامی بود؟

ویرث: خوب، من هرگز نیازی به پشـــتهٔ دوم حـــس نکردم و ضمناً همیشه هم به دنبال راه‌حل‌های ساده‌تر هستم.

دِی‌لایت: علّت این پرســـش من این بود که شما بعداً برای Lilith (نام یک ایســـتگاه کار[۲۲] که با اســـتفاده از پردازندهٔ AMD ۲۰۹۱ در دانشـــگاه فنی زوریخ توســـط **ویرث** و همکارانش ساخته شد. این پروژه در ســـال ۱۹۷۷ آغاز شـــد و تا ســـال ۱۹۸۴ چند صد ایســـتگاه کار در حال استفاده بودند. مترجم) از دو پشته استفاده کردید[۲۷۰].

ویرث: بله، امّا آن نوع متفاوتی از پشته بود. در آنجا یک پشتهٔ معمولی برای رکوردهای فعّال‌ســـازی رویّه و یک پشـــتهٔ اضافی برای محاسبهٔ عبارت‌های جبری وجود داشـــت. پشـــتهٔ دوم، به حکم سخت‌افزار، به صورت ۱۶ ثبّات[۲۳]، پیاده‌ســـازی شـــده بود. می‌توانید آن را به‌صورت توسعهٔ پشتهٔ عادی در نظر بگیرید.

دِی‌لایت: این تلاش و کوشـــش برای ســـادگی، بســـیار جالب است. دایکســـترا هم فقط یک پشـــته داشـــت. امّا هنگامی که به کارهای دیگران نگاه کنیم می‌بینیم که اغلب از چند پشـــتهٔ تخصصی استفاده کرده‌اند. حتی از هفت پشـــته در ماشـــین مجازی SDL، به خاطر بالا بردن کارایی، و البته به بهای از دســـت دادن ســـادگی، استفاده شده اســـت[۲۷۰]. هنگامی که تلاش می‌کردید از تنها یک پشته استفاده کنید، آیا به فکر کارایی هم بودید؟

ویرث: (با خنده) من همیشه یک معلّم بوده‌ام و شما باید به‌عنوان یک معلّم، چیزهایی را برای افراد توضیح دهید. اگر چیزی که می‌خواهید درباره‌اش توضیح دهید به قدر کافی ســـاده باشـــد، خیلی خوشحال خواهید شد زیرا آن‌ها توضیحات شما را به راحتی درک خواهند کرد. مـــن این را بارها و بارها تجربه کرده‌ام، حتی در برنامه‌نویســـی، مردم معمولاً برنامه را برای ماشـــین می‌نویسند تا بر روی آن اجرا شود. من فکر می‌کنم در دانشـــگاه‌ها افراد باید طوری برنامه بنویسند که برای دیگران آموزنده باشد. این البته یک ایدهٔ دانشگاهی است. در دانشگاه فنّی زوریخ نیز اغلب افراد نمی‌توانند طوری برنامه بنویســـند که من آن را درک کنم یا برای دیگران قابل توضیح‌دادن باشد. بنابراین، من همیشه حس کرده‌ام که باید ماشین‌ها، زبان‌ها و سیستم‌عامل‌هایمان را تا حدّ امکان ساده درست کنیم.

دِی‌لایت: در این صورت من می‌توانم تصوّر کنم که هنگامی که برای نخستین بار دایکسترا را دیدید، خود را با طرز فکر او هم‌راستا یافتید. آیا با او در مونیخ در سال ۱۹۶۲ صحبت کردید؟

ویرث: بله، خیلی زیاد. همچنین البته از نوشته‌هایش. بله، من او را در مونیخ ملاقات کردم امّا ملاقاتمان خیلی کوتاه و مختصر بود.کار من بر روی زبان اویلر البته برای وَن‌واین‌گاردن شناخته شده بود و به همین دلیل از من برای مشارکت در گروه کاری IFIP دعوت کرد. من دعوت او را پذیرفتم و برای نخســـتین بار در جلسهٔ گروه در مارچ ۱۹۶۴ در شهر توتزینگ در نزدیکی مونیخ شرکت کردم.

دِی‌لایت: در آنجا تونی هور را هم برای نخستین بار دیدید؟

21- portable

22- workstation

23- register

ویرث: بله، دقیقاً.

۵-۱-۴ معناشناسی

دِی‌لایت: در مقالهٔ «اویلر: تعمیمی از الگول و تعریف صوری آن: بخش اوّل»[۳۱۲]، برخی از رویکردهای متفاوتی را که افراد مختلف به منظور تعریف معناشناسی زبان برنامه‌نویسی در پیش گرفته‌اند، توضیح داده‌اید. شما به بوهم[۲۲] و لَندین[۱۶۵,۱۶۶] اشاره کرده‌اید که به مقدار زیادی از حساب لاندا[۲۴] الهام گرفته‌اند. شما همچنین رویکرد پویای وَن‌واین‌گاردن و رویکرد گارویک را توضیح داده‌اید.

ویرث: کجا من این کار را کرده‌ام؟

دِی‌لایت: در این مقاله[۳۱۲] (دِی لایت مقاله را نشان می‌دهد).

ویرث: آها، خوب (با خنده) مال خیلی وقت پیش است.

دِی‌لایت: مرور بسیار جالبی است. شما مثلاً به کاربست‌پذیری[۲۵] و نحوهٔ ارزیابی در رویکرد وَن‌واین‌گاردن اشاره کرده‌اید.

ویرث: شما در من این انگیزه را به وجود آوردید که کارهای پیشینم را دوباره بخوانم. (خندهٔ هر دو)

دِی‌لایت: و بعد گارویک که به نظر می‌رسد رویکردی کاملاً عملیاتی را پیشنهاد کرده است.

ویرث: بله، کاملاً. من گارویک را خیلی خوب به یاد می‌آورم، آدمی پرانرژی که خیلی به تعریف ماشین‌انگارانه‌اش وفادار بود. با تجربه‌ای که من در توضیح یک زبان برحسب یک زبان ساده‌تر داشتم، حس می‌کردم که این پایان داستان نیست. بدین خاطر، رویکرد اصل موضوعی که فکر می‌کنم فلوید و بعداً هور اشاعه دادند، بسیار نویدبخش‌تر بود.

دِی‌لایت: این بعد از کار شما بر روی اویلر بود؟

ویرث: بله، کار اویلر بین ۱۹۶۳ و ۱۹۶۶ انجام گرفت.

دِی‌لایت: فلوید و هور رویکرد کاملاً متفاوتی در مقایسه با حساب لاندا داشتند. شما برای اصل موضوعی‌کردن پاسکال، از نزدیک با هور کار کردید. درست است؟

ویرث: بله. من با هور دربارهٔ تعاریف اصل موضوعیش گفتگو کرده بودم و آن را رویکرد بسیار بهتری یافته بودم. به او پیشنهاد کردم که روشش را بر روی پاسکال اعمال کنیم و بدین ترتیب، با هم نشستیم، با هم کار کردیم و مقاله‌ای دربارهٔ تعریف اصل موضوعی پاسکال نوشتیم. من همین دیروز دنبالش گشتم امّا نتوانستم پیدایش کنم. راستش را بخواهم به شما بگویم، ما واقعاً کار را تمام نکردیم. بعضی قسمت‌ها بود که واقعاً نمی‌دانستیم با آن چگونه رفتار کنیم، به ویژه پارامترهای var و مسئلهٔ نام‌های ساختگی[۲۶]. gotoها هم چیز بدی بودند. و نمی‌توانستید به‌صورت اصل موضوعی goto را تعریف کنید مگر آن که پیشرفت مهمی صورت می‌گرفت. در حالی که حلقه‌ها، مثلاً دستور while، تعریف اصل موضوعی خیلی زیبایی داشتند.

24 -calculus
25- applicability
26- alias

بنابراین، در زبان‌های بعدی که من طراحی کردم، اصلاً goto وجود نداشت. (خندهٔ **ویرث**)

دِی‌لایت: بدین ترتیب، صورت‌بندی شما را مجبور کرد که مفاهیم خوب را حفظ کنید.

ویرث: و آن‌هایی که در آن قالب جا نمی‌گرفتند را بیرون بریزم.

دِی‌لایت: نظرتان در مورد حساب لاندا چیست؟ عدّه‌ای بسیار علاقه- و می‌توانم تقریباً بگویم وسواس- دارند که از آن استفاده کنند. این علاقه‌مندی از کجا می‌آید؟ آیا انگیزهٔ عملی برای این کار دارند یا عمدتاً به خاطر علاقه‌مندی به آراستگی و زیبایی ریاضی است؟

ویرث: این علاقه‌مندی از سوی کسانی بوده است که گرایش قوی ریاضی داشته‌اند. از جان‌مک‌کارتی سرچشمه می‌گیرد و نیز مقاله‌ای در کنگرهٔ IFIP در مونیخ در سال ۱۹۶۲، که مقالهٔ بسیار جالبی بود با عنوان «به سوی علم ریاضیِ محاسبات»[۱۸۶]. در آن هیچ اشاره‌ای به خود رایانه‌ها نشده است. امّا چون همه چیز بر حسب فراخوانی‌های تابع از طریق بازگشت[۲۷] تعریف شده، پرسشی که پیش می‌آید این است که: چه چیزی به دست می‌آوریم؟ برای برنامه‌های ساده خوب است امّا برای مسائل پیچیده، نه. مقالهٔ مک‌کارتی، سرآغاز مکتب زبان‌های برنامه‌نویسی تابعی[۲۸] است. عجیب آن که این مکتب در انگلستان، و عمدتاً در اسکاتلند، جا باز کرد. ادینبورگ سرچشمهٔ زبان‌های برنامه‌نویسی تابعی است. نام‌هایی که به یادم می‌آید میلنر و استراچی هستند.

۵-۱-۵ بازگشت

دِی‌لایت: شما به بازگشت اشاره کردید. این موضوع در تعریف الگول ۶۰ بحث‌انگیز بود. یک پرسش احساسی در اوایل دههٔ ۱۹۶۰ این بود که آیا برنامه‌های مترجم (کامپایلر) الگول ۶۰ باید بازگشت را پیاده‌سازی کنند یا نه. در واقع، شما خودتان نیز دربارهٔ پاسکال به آن اشاره کرده‌اید:

«منع استفاده از بازگشت برطرف شده بود. بازگشتی‌بودن رویّه‌ها عادی تلقّی می‌شد.»[۳۰۷]

ویرث: ببینید، من به‌عنوان یک مهندس، آموزش دیده بودم. حس می‌کردم که پاسکال باید به نحوی پیاده‌سازی شود که مردم علاقه‌مند به استفاده از آن باشند. این یعنی این که باید پیاده‌سازی کارآمدی داشته باشد. یادم می‌آید که برای سال‌های طولانی، تا اواخر دههٔ ۱۹۷۰، فرترن الگوی پیاده‌سازی کارآمد بود و من می‌دانستم که یک زبان دیگر، تنها هنگامی مورد پذیرش گسترده قرار خواهد گرفت که برنامه‌هایش به همان کارآمدی برنامه‌های فرترن اجرا شوند. یکی از موانع عمده بر سر راه کارآیی، بازگشت بود زیرا نیازمند یک پشته بود. کسانی که فقط به زبان فرترن برنامه‌نویسی می‌کردند، هیچ مزیتی در استفاده از پشته نمی‌دیدند. بعداً، به‌ویژه هنگام نوشتن برنامه‌های مترجم با در نظر گرفتن بازگشت، به این نتیجه رسیدم که بازگشت

27- recursion
28- functional programming

در واقع یک ویژگی خیلی اساسی بود. در آن زمان، همچنین یاد گرفته بودم که پیاده‌سازی آن زیاد هم پیچیده و مشکل نیست، به‌ویژه آن که رایانه‌هایی با قابلیت نشانی‌دهی مبنا[29] از طریق ثبّات‌ها، در دسترس قرار گرفته بودند.

دِی‌لایت: در مورد الگول ۶۰، دایکسترا و زونِولد یکی از نخستین کسانی بودند که بازگشت را در برنامهٔ مترجمشان پیاده‌سازی کردند. در آگوست ۱۹۶۰، برنامهٔ مترجم آن‌ها به‌طور واقعی کار می‌کرد. این مسئله، تعجّب پیتر نور و شاید بتوانم بگویم تمام اروپا را برانگیخت. برای نمونه، برخی پژوهشگران انگلیسی به آمستردام رفتند تا از دایکسترا بیاموزند که چگونه می‌توان یک برنامهٔ مترجم الگول ۶۰ نوشت که بتواند بازگشت را مدیریت کند. دایکسترا بازگشت را با یک پشتهٔ زمان اجرا پیاده‌سازی کرده بود. امّا خودش نیز معترف بود که برنامه‌های الگول ۶۰ او از نظر زمان اجرا کارایی بالایی نداشتند. او در انتظار روزی بود که سرعت رایانه‌ها به قدر کافی زیاد شود. من فکر می‌کنم که او به پشتهٔ سخت‌افزاری اشاره می‌کرد. دایکسترا می‌گفت «ممکن است امروز کارایی نداشته باشد امّا در آینده چنین خواهد شد.» او تمام این چیزها را در سخنرانی مهمش[۶۴] در مونیخ در سال ۱۹۶۲ توضیح داد.

ویرث: بله، ماشین‌ها در آن زمان برای زبان‌های خوب، واقعاً مناسب نبودند. آن‌ها برای فرترن ساخته شده بودند. (خندهٔ **ویرث**) امّا بعداً ماشین‌ها قوی‌تر شدند، به ویژه در سال ۱۹۶۴ با به بازار آمدن سیستم ۳۶۰ آی‌بی‌ام که آرایه‌ای از ثبّات‌ها داشت. از همهٔ آن‌ها می‌شد به‌عنوان ثبّات‌های شاخص[30] و ثبّات‌های مبنا[31] استفاده کرد و بدین ترتیب در به روی پیاده‌سازی کارآمد بازگشت گشوده شد.

دِی‌لایت: آیا فقط به خاطر کارایی بود که بازگشت را نمی‌خواستند؟ یا این که درک هم نمی‌کردند که چگونه می‌تواند در برنامه‌نویسی مفید باشد؟

ویرث: بله، دقیقاً.

دِی‌لایت: باوئر و ساملسون در برنامه‌های مترجم ALCOR خود، بازگشت را پیاده‌سازی نکرده‌اند.

ویرث: درست است.

دِی‌لایت: آن‌ها خیلی نسبت به این مسئله حسّاس بودند که آمستردامی‌ها داشتند این کار را انجام می‌دادند. این قضیه نشان می‌دهد که این فقط یک بحث فنّی نبوده است.

ویرث: خیلی تعجّب‌آور است زیرا باوئر ریاضی‌دان نیز بود. البته اگر دربارهٔ جبر خطی، محاسبات ماتریسی و چیزهایی از این قبیل صحبت کنید، بازگشت واقعاً اهمیت ندارد. امّا البته با قوی‌تر شدن رایانه‌ها و مهم‌تر شدن آن‌ها، کاربردهای بیشتر و بیشتری مطرح شدند که بازگشت برای آن‌ها مفید بود و برنامه‌های مترجم از اولین آن‌ها بودند. ضمناً این را هم بگویم که فکر می‌کنم این روش پیاده‌سازی بازگشت، نخستین بار توسط هور مورد استفاده قرار گرفت.

29- base addressing
30- index registers
31- base register

دِی‌لایت: همچنین گراو[۱۱۱] و آیرونز[۱۴۷].

ویرث: گراو، شاید. و بعد این ایدهٔ جدید: تجزیهٔ پایین به بالا[32] به جای بالا به پایین. من به کار کردن بر روی ایدهٔ دستور زبان‌های تقدّمی[33] فلوید ادامه دادم و یک دلیلش این بود که دیگر به بازگشت نیاز نداشتید: همه چیز، جدولی از داده‌های نحوی بود و یک مفسّر بر روی آن اجرا می‌شد. امّا پس از برنامهٔ مترجم الگولW، من تمام آن کارها را متوقف کردم و دوباره سراغ بازگشت رفتم که بسیار شفّاف‌تر است.

دِی‌لایت: آیا همکاری نزدیکی با فلوید داشتید؟

ویرث: نه، فکر می‌کنم او در پیتسبورگ بود. کنوث یکسال پس از آن که من از استنفورد رفتم، و شاید هم در همان سال، به آنجا آمد و شرط‌پذیرش استادی در آنجا را این قرارداد که فلوید هم به‌عنوان استاد منصوب گردد. بدین ترتیب، فلوید یکسال پس از آن که من از استنفورد رفتم به آنجا آمد.

۵-۲ بازنگری دههٔ ۱۹۷۰

دِی‌لایت: در خلال دههٔ ۱۹۷۰، فناوری الکترونیک پیشرفت چشمگیری داشت. در پایان این دهه، کانون توجّه دانشگاهیان و متخصصان تغییر عمده‌ای یافت: توسعه‌پذیری و قابلیت نگهداری[34] خیلی اهمیت پیدا کرد و از اهمیت کارایی ماشین کاسته شد. شما در مقالهٔ ۱۹۸۸ خود بر این نکته تأکید کرده‌اید:

«برنامه‌نویسی عبارت است از توسعهٔ یک سیستم مفروض.
توسعه‌پذیری، سنگ‌بنای تولید سیستم است زیرا به ما امکان می‌دهد تا سیستم‌های جدید را بر پایهٔ سیستم‌های موجود بنا کنیم و هر کار را دوباره از نقطهٔ صفر آغاز نکنیم.»[۳۰۵]

برای بحث در این باره، پیشنهاد می‌کنم که با «بحران نرم‌افزار» که در همایش ۱۹۶۸ در شهر گارمیش آلمان مطرح شد آغاز کنیم. آیا شرکت‌کنندگان در آن همایش، مثلاً خود شما، واقعاً در آن سال بحران را حس کرده بودند؟

۵-۲-۱ بحران نرم‌افزار -۱۹۶۸

ویرث: خوب، پیش از هر چیز باید بگویم که من در آن همایش شرکت نداشتم. امّا یک حس قوی وجود داشت که باید کاری برای مقابله با این پیچیدگی روزافزون و ناتوانی در تولید سیستم‌های بدون خطا، صورت گیرد. به یاد داشته باشید که این زمانی بود که شرکت‌های بزرگ تلاش می‌کردند سیستم‌های اشتراک زمانی[35] را پیاده‌سازی کنند در حالی که ماشین‌های بزرگ فقط برای پردازش دسته‌ای[36] به‌کار گرفته می‌شدند. ایدهٔ اشتراک زمانی، یعنی سیستم‌های چندپردازی[37] و چند کاربره، ابتدا توسط مک‌کارتی مطرح گردید. شرکت‌ها این ایده را قاپیدند و خیلی پیش از آن که به‌طور واقعی

32- bottom-up parsing
33- precedence grammars
34- maintainability
35- time-sharing
36- batch processing
37- multi-processing

چنین سیســـتم‌هایی را پیاده‌سازی کرده باشند، به آگهی‌کردن آن‌ها پرداختند. در نتیجه، پیاده‌ســـازها زیر فشـــار شدیدی قرار گرفتند در حالی که هنوز برای مقابله با مشـــکلات ذاتی عملکرد چندبرنامه‌ای[38] آمادگی نداشـــتند. این یک بحران واقعی بود. بـــرای نمونه، آی‌بی‌ام هـــزاران برنامه‌نویـــس را فقط به کار بر روی سیســـتم عامل OS/360 گماشـــته بود. من فکر می‌کنم ارزش همایشـــی که شما به آن اشاره کردید این بود که این واژهٔ بحران را به میان انداخت و آشکار ساخت. نخستین بار بود که افراد سطح بالای این شرکت‌ها اعتراف کردند که با مشـــکلات عمیقی دست به گریبان هستند، زیرا پیش از آن همواره این موضوع را پنهان نگاه می‌داشتند. آن‌ها نمی‌خواستند مشتریانشان را از دست بدهند، مشتریانی که وعده‌های بزرگی بهشان داده بودند. بنابراین، بله یک مشکل واقعی بود.

دِی‌لایت: در همایش بعدی در ســـال ۱۹۶۹، دایکسترا ایده‌های خود دربارهٔ برنامه‌نویسی ســـاخت‌یافته[39] را ارائه کرد[۷۱]. این واکنشی به بحران نرم‌افزار بود، مگر نه؟

ویرث: بله، مثل همیشـــه، دایکســـترا جلوتر از جمعیت بود. (خندهٔ **ویرث**) این‌طور که یادم می‌آید، سال ۱۹۶۵ بود که من یک نسخه از نوشتهٔ او با عنوان «یادداشت‌هایی دربارهٔ برنامه‌نویسی ساخت‌یافته» را به دست آوردم. نوشتهٔ روشنگری برای من بود. منظورم این است که من خودم نیز تمام آن ایده‌ها را داشـــتم امّا آن‌ها را به شکل منظّمی بر روی کاغذ نیاورده بودم. به نوعی می‌توان گفت پاسکال واکنش من به آن بود. به خودم گفتم «این ایده‌ها عالی هســـتند امّا باید آن‌ها را در دســـترس برنامه‌نویسان قرار داد، و نه فقط به شکل مقاله، بلکه در قالب زبانی که این نظام را در برداشته باشد.» ایدهٔ پاسکال چنین بود.

دِی‌لایت: پاســـکال فقط دربردارندهٔ برنامه‌نویسی ساخت‌یافته نبود، بلکه تقاضای هور برای معناشناسی اصل موضوعی را نیز برآورده کرده بود.

ویرث: بله، رویکرد هور فقط در زبانی قابل به کاربستن بود که ساختار عبارتی[40] خوبی داشت و پاسکال برای آن مناسب بود.

۵-۲-۲ صورت‌بندی‌ها[41]: گذشته و حال

دِی‌لایت: به بحث در مـــورد رویکرد اصل موضوعی هور ادامه دهیم. شما گفته‌اید:

«دههٔ ۱۹۷۰ ســـال‌هایی بـــود که همه عقیده داشـــتند اثبات صحّت صوری[42] برنامه‌ها، هدف نهایی اســـت. هور اصول موضوع و قاعده‌های اســـتنتاج دربارهٔ نشان‌گذاری‌های برنامه‌نویسی (که بعداً به نام منطق هور معروف شد) را اصل قرار داده بود. او و من کار تعریف معناشناسی پاســـکال را با به‌کارگیری این منطق انجـــام دادیم. با وجود این، باید اذعان کنیم که تعدادی از ویژگی‌ها (مانند اشاره‌گرها[43]) باید از تعریف صوری حذف می‌شدند.»[۳۰۷]

به نظرم می‌آید که شما واقعاً می‌خواستید این رویکرد اصل موضوعی را امتحان کنید. امّا شما در مقالهٔ اخیرتان نوشته‌اید:

«نهایتـــاً قراربود وارســـی تحلیلی و اثبات صحّـــت برنامه‌ها جایگزین آزمایش[44] گردد، امّا این اتفاق نیفتاد.»[۳۱۱]

من فکر می‌کنم شـــما تردیدهایی نیز دربارهٔ زبان‌های برنامه‌نویســـی که توســـط مســـتندات خیلی طولانی به‌طور صوری معرفی شده‌اند دارید. آیا این درست است که شما امروز دیدگاه معتدل‌تری نسبت به روش‌های صوری پیدا کرده‌اید؟

ویرث: کاملاً. دو هفتـــهٔ دیگر در اینجا همایش کوچکی برگزار خواهد شـــد و افراد مختلفی از جاهای مختلف دنیا که در زمینهٔ اثبات صحّت برنامه‌ها کار کرده‌اند گرد می‌آیند تا ۶۰ ســـالگی برتراند مِیر را جشن بگیرند. از من خواسته‌اند که برایشان سخنرانی کنم و من هم پذیرفته‌ام. و علتش هم این بود که فرصتی به دست خواهم آورد تا نگرانی‌هایم را بیان کنم. (خندهٔ ویرث) آن‌ها غرق در فضای دانشگاهی و عاشق مفاهیم ریاضی هستند. اکنون اغلب برنامه‌نویسان، ریاضی‌دان نیستند. ما با این روش‌های صوری، به جای آن که موانعی را از سر راهشان برداریم، موانع و مشکلات بیشتری را سر راهشان قرار می‌دهیم. اثبات صحّت برنامه‌ها و این روش‌های تحلیلی (جملات اظهاری[45] هور یا مبدّل‌های محمول[46] دایکسترا) برای برخی مثال‌های کوچک خوب کار می‌کنند امّا نه برای سیســـتم‌های بزرگ. شـــما هنوز می‌توانید از جملات اظهاری استفاده کنید امّا اثبات صوری آن‌ها داستان دیگری است. من فکر می‌کنم برای آموزش برنامه‌نویسی خوب است و داشتن اثبات به همراه طراحی، ایدهٔ مهمی است. امّا شیوه‌ای که نظریه‌پردازان آن را به وجود آورده‌اند شبیه یک ابر نظری اســـت که از جایی که قرار بوده باشد، حرکت کرده و به جای دیگری رفته است.

دِی‌لایت: خیلی جالب اســـت زیرا دایکســـترا در دههٔ ۱۹۶۰ چنین کارهـــای صوری انجام نمـــی‌داد. او بعد از هور و دیگـــران بود که به رویکرد صوری اعتقاد پیدا کـــرد. امّا در نیمهٔ دوم فعّالیت حرفه‌ایش، کارهای او کاملاً صوری بود.

ویرث: بله، و بگذارید بگویم که من از این بابت متأســـفم. او دنیای برنامه‌نویسان را ترک کرد. در دهسال پایان عمرش، هیچ فعالیتی در دانش رایانه (علوم کامپیوتر) نکرد و صرفاً به ریاضیات و مسائل ریاضی پرداخـــت. دنیای رایانش هر روز فاصلهٔ بیشـــتری با او حس می‌کرد و دیگر او را جدّی نمی‌گرفت. این خیلی بد امّا قابل درک بود.

دِی‌لایت: می‌خواهم این کتاب را به شما نشان دهم: «مکتب نیکلاوس **ویرث**: هنرِ سادگی»[۲۴]. (خندهٔ دِی‌لایت و ویرث)

ویرث: واقعاً عالی بود.

38- multiprogramming
39- structured programming
40- phrase structure
41- formalism
42- formal proof of correctness
43- pointers
44- testing
45- assertion
46- predicate transformer

دِی‌لایت: و هنگامی که به فصل دایکسترا در این کتاب که به خط خود اوست نگاه می‌کنم، می‌بینم خیلی صوری است (خندهٔ هر دو)
ویرث: خیلی جالب است، مگر نه؟ (خندهٔ هر دو)
دِی‌لایت: به نوشته‌های دیگران در این کتاب هیچ شباهتی ندارد.
ویرث: نه، و این به خاطر این که نوشتهٔ او دستخط خودش است نیست. (خندهٔ هر دو)

۵-۲-۳ دانشگاهیان پسامدرن[47]

دِی‌لایت: این تغییر کانون توجه در خلال دههٔ ۱۹۷۰ به خاطر پیشرفت چشمگیر فناوری الکترونیک بود. امروز نیز کارایی در مرتبهٔ دوم قرار دارد، در حالی که صحّت و سادگی جایگاه نخست را دارند. درست است؟
ویرث: شما می‌توانید این گونه فکر کنید، امّا آیا در عمل هم این چنین است؟ نمی‌دانم.
دِی‌لایت: خیلی خوب، منظورم در محیط‌های دانشگاهی است.
ویرث: بله، حق با شماست. امّا مگر چند نفر از استادان دانش رایانه هنوز برنامه می‌نویسند؟ خیلی کم. البته آن‌ها می‌گویند «من خودم می‌توانم این کار را بکنم امّا آن را بر عهدهٔ دستیارم گذاشته‌ام و نیازی نیست خودم انجام دهم»، که البته کاملاً اشتباه است. زیرا تنها هنگامی که خودتان انجام دهید، و منظورم مثال‌های کوچک دایکسترا نیست بلکه سیستم‌های واقعی (برنامه‌های مترجم، سیستم‌های عامل)، متوجه خواهید شد که چه مشکلاتی وجود دارد. این واقعیت که استادان خودشان این کار را انجام نمی‌دهند، نتیجه‌اش این می‌شود که ارتباطشان با مسئله را از دست می‌دهند و دیگر چیزهای مهم را تدریس نمی‌کنند. آن‌ها به سمت چیزهای نظری (که فعالیت ذهنی لذت‌بخشی است) روی می‌آورند، در حالی که پارناس نوشته است که دانشجویانی که جذب صنعت می‌شوند مهندسان هستند. آیا مقالهٔ او را خوانده‌اید؟ مقالهٔ «آموزش برای حرفه‌ای‌های رایانش»؟[۲۳۵]
دِی‌لایت: نه، این مقاله را نخوانده‌ام امّا بسیاری از مقالات او را خوانده‌ام. مایلم به آنچه شما گفتید چیزی را بیفزایم. شما دربارهٔ دانشگاهیانی که دیگر برنامه نمی‌نویسند، و مشخصاً سیستم‌های عامل و برنامه‌های مترجم، صحبت می‌کنید. هنگامی که من به آموزش خودم به‌عنوان مهندس نرم‌افزار نگاه می‌کنم، می‌بینم که هیچگاه لازم نبوده است که یک سیستم عامل یا برنامهٔ مترجم کوچک بنویسم. از یک طرف، این کاملاً قابل درک است و حتی می‌توان به آن به‌عنوان یک پیشرفت علمی نگریست: کنار گذاشتن جزئیات سطح پایین به ما اجازه می‌دهد که منحصراً بر روی دامنهٔ کاربرد[48] تمرکز کنیم. از سوی دیگر، دستاوردهای برنامه‌نویسان نسل اوّل، مانند مشارکت‌های اوّلیهٔ دایکسترا، غالباً توسط نسل‌های جدید برنامه‌نویسان درک نمی‌شود.
ویرث: درست است. ما نمی‌توانیم این روند را تغییر دهیم. امّا حداقل می‌توانیم کاری در مقابلش انجام دهیم. مثلاً من برای چندین سال درس طراحی برنامهٔ مترجم (کامپایلر) را تدریس کرده‌ام و هیچگاه از این کتاب‌های قطور درسی که شامل هر چیز که فکرش را بکنید دربارهٔ برنامه‌های مترجم هستند، استفاده نکرده‌ام. به جای آن، زبانی به نام Pascal-0 ساخته‌ام که اغلب ساخت‌های پایه را دارا می‌باشد و بعد دانشجویان را وا داشته‌ام که برنامهٔ مترجمی برای این زبان بنویسند. من اسکلت پودمان‌ها[49] را به آن‌ها داده‌ام و ابتدا از آن‌ها خواسته‌ام که فقط تحلیل نحوی[50] را انجام دهند. سپس به سراغ پردازش اعلان‌ها[51]، تشکیل ساختمان داده‌ها و نمایش متغیرها و این چیزها رفته‌ایم. و در نهایت، تولید کد. در پایان ۱۲ یا ۱۴ هفته، آن‌ها موفق شده‌اند که برنامهٔ مترجم کوچک خود را بسازند. آن‌ها همیشه راضی بوده‌اند و می‌گفتند که «حالا فهمیدیم». آن‌ها می‌توانستند وارد صنعت شوند و برنامه‌های مترجم پیچیده‌ای را برای زبان‌های پیچیده بنویسند. آن‌ها چیزهای پایه و اساسی را یاد گرفته بودند، چیزهایی که شما فقط با خواندن کتاب یاد نمی‌گیرید. من فکر می‌کنم باید هر چه بیشتر به این شیوه تدریس کرد، امّا این وقتی امکان‌پذیر است که معلّم خودش این کار را انجام داده باشد. من همیشه خودم ابتدا کاری را انجام می‌دهم و بعد آن را تدریس می‌کنم. در غیر این‌صورت، خیلی احساس عدم اعتماد به نفس خواهم داشت.
دِی‌لایت: شما در نوشته‌هایتان از عبارت «پَسامدرن» نیز استفاده کرده‌اید:
«در این محیط دانشگاهی پَسامدرن، استاد از مدّت‌ها پیش به‌عنوان آدم آگاه وخردمندی شناخته شده که عمیق‌تر و عمیق‌تر به موضوع مورد علاقه‌اش پرداخته و مطالعه کرده است. استاد مدرن، مدیر یک تیم پژوهشی بزرگ است که سخت به دنبال برقراری ارتباط نزدیک با بنگاه‌های سرمایه‌گذاری است تا بودجهٔ پژوهش‌هایش را تأمین کند و به‌طور خستگی ناپذیری به نوشتن پیشنهادهای[52] هیجان‌انگیز پروژه و داستان‌های حیرت‌انگیز موفقیت می‌پردازد.»[۳۱۰]
امّا این بلافاصله به ما می‌گوید که اگر دانشگاه بخواهد بر روی سیستم‌های عامل کار کند، به سختی خواهد توانست آن را به عنوان یک چیز جدید بفروشد. واکنش عادی این خواهد بود که بگویند «ما تمام چیزهایی که باید دربارهٔ سیستم‌های عامل بدانیم را می‌دانیم، به این کاری نداشته باش، کار دیگری انجام بده».
ویرث: درست است. ما نوآوری‌های زیادی در رشته‌مان داریم امّا کار کمی برای رسیدن به کمال انجام می‌دهیم. مهندسان نمی‌توانند کاری را انجام دهند و در همان بار اوّل همه چیز عالی و درست باشد. آن‌ها معمولاً کاری را انجام می‌دهند و سپس آن را بهبود می‌بخشند و تنظیم می‌کنند. و این کار در نرم‌افزار انجام نشده است و برنامه‌نویسان فقط باید بیشتر و بیشتر کد بنویسند. و در آخر،

47- post-modern
48- application domain
49- modules
50- syntax analysis
51- declaration
52- proposal

هیچکس نمی‌داند چقدر نرم‌افزار وجود دارد، مگر رایانه‌ها!

دِی‌لایت: تونی هور مطلبی گفت که به نظر من کمی طعنه‌آمیز است. او می‌گوید در مرکز پژوهشی مایکروسافت و در نتیجه در صنعت، آزادی عمل بیشتری در مقایسه با هنگامی که در دانشگاه بود، دارد. البته این تا حدودی به خاطر موقعیتی است که او دارد. امّا واقعیت این است که مایکروسافت و گوگل پول بیشتری برای پژوهش‌های بنیادی دارند تا دانشگاه‌ها و مراکز علمی.

ویرث: خیلی بیشتر. آن‌ها تنهاجاهایی هستند که می‌توانند چنین امکاناتی را برای پژوهشگران برجسته فراهم کنند و به آن‌ها بگویند «هر کاری دوست دارید انجام دهید، از نظر ما اهمیتی ندارد. ما ازشهرت شما آگاهیم و می‌دانیم کار خوبی انجام خواهید داد.» دانشگاه‌ها دیگر این‌گونه عمل نمی‌کنند. از یک استاد انتظار می‌رود که بیش از پولی که می‌گیرد، پول به دانشگاه بیاورد.

دِی‌لایت: دانشگاه‌ها امروز باید بر روی آنچه می‌فروشند تمرکز کنند. به گفتهٔ خود شما:

«دانشگاه‌ها به‌طور سنتی از این‌گونه تمرکزهای تجاری برکنار بوده‌اند. دانشگاه‌ها جایی بودند که از افراد انتظار می‌رفت دربارهٔ چیزهایی که در بلند مدّت اهمیت پیدا خواهند کرد، ژرف‌اندیشی کنند.»[۳۱۱]

ذکر این نکته جالب است که در خلال دهه‌های ۱۹۴۰ و ۱۹۵۰، رایانه‌ها در دانشگاه‌ها ساخته می‌شدند. و بعداً تولید رایانه به‌طور کامل به صنعت منتقل شد. در خلال دهه‌های ۱۹۵۰ و ۱۹۶۰، چند زبان برنامه‌نویسی جدید در دانشگاه‌ها طراحی شدند. و بعد صنعت، زبان‌های مورد علاقهٔ خود مانند C، C++ و جاوا را برگزید. در نتیجه، دانشگاه تا حدّ زیادی طراحی زبان‌های جدید را متوقف ساخت. دانشگاه اکنون برای دستیابی به بودجه‌های پژوهشی، نیازمند کار با این زبان‌هاست، حتی با وجودی که زبان‌های بسیار بهتری نیز موجود می‌باشند. و این مطلب را شما هم در نوشته‌هایتان ذکر کرده‌اید. من نمی‌دانم این روند چگونه ادامه خواهد یافت. شاید یک دانشور واقعی، که می‌خواهد کتاب هم بخواند- در واقع، اغلب استادان حتی وقت کتاب‌خواندن هم دیگر ندارند- باید کار در دانشگاه را رها کند. به نظر نمی‌رسد تلاش برای مقابله با این جریان غالب، ایدهٔ خوبی باشد.

ویرث: بله، غیرممکن شده است. زبان‌هایی مانند C در صنعت درست شدند، در مراکز پژوهشی AT&T. همچنین C++ نیز آنجا درست شد. جاوا در شرکتِ سان و C# در مایکروسافت. و حالا زبان Go در گوگل. هر کس می‌خواهد زبان خودش را داشته باشد، با وجودی که آن‌ها بسیار شبیه هم هستند. به نظر می‌رسد که زمان تعریف زبان دیگر به سرآمده است. همه وقتی که صحبت از زبان می‌کنند، عمدتاً دربارهٔ نحو صحبت می‌کنند و این هم زبان C است، چه دوستش داشته باشید چه نداشته باشید. به محض آن که چیزی خود را پابرجا کرد، مقابله با آن بسیار دشوار می‌گردد. شما باید دلایل بسیار قوی و محکمی داشته باشید. این مسئله در مورد معماری‌های رایانه نیز صادق است. اینتل نخستین شرکتی بود که ۸۰۸۰ را بیرون داد. موتورولا ۶۸۰۰ فقط شش ماه بعد از آن عرضه شد که واقعاً خیلی بهتر هم بود. بعد اینتل ۸۰۸۶ را بیرون داد و فکر می‌کنم باز هم شش ماه بعد از آن موتورولا ۶۸۰۰۰ عرضه شد که طراحی بسیار بهتری داشت. امّا هیچکس دیگر دربارهٔ موتورولا صحبت نمی‌کند. اینتل تنها چیزی است که هنوز هم وجود دارد. بنابراین، از یک طرف بی‌تحرّکی و بی‌حالی زیادی در جامعهٔ کاربران وجود دارد و از سوی دیگر، نخستین چیزی را که پابرجا شد به سختی می‌توان از میان برداشت.

دِی‌لایت: بنابراین، زمان برای آن که دانشگاه به تعریف زبان‌های برنامه‌نویسی جدید بپردازد، سپری شده است.

ویرث: بله، حتی با وجودی که به شدّت به زبان‌های برنامه‌نویسی بهتر نیاز هست، این اتفاق نمی‌افتد. پاسکال، گام مهمی به جلو نسبت به فرترن بود. الگول بجز برخی جاها در اروپا، شناخته شده نبود و هر چیز که بعد از آن آمد، گام کوچکی به جلو بود. برای معرفی یک چیز تازه، گامی که به جلو برداشته می شود باید آنقدر بزرگ باشد که توجیه‌کنندهٔ تغییر باشد.

۵-۲-۴ نرم‌افزار فربه

دِی‌لایت: اتفاق دیگری که از دههٔ ۱۹۶۰ افتاده این است:

«برنامه‌ها و سیستم‌ها به نحو فزاینده‌ای پیچیده شده‌اند و تقریباً غیرممکن است که یک نفر به تنهایی به‌طور کامل آن‌ها را درک کند.»[۳۱۱]

علاوه بر این، شما گفته‌اید که:

«محدودیت‌های ما در طراحی سیستم‌های پیچیده، دیگر نه توسط سخت‌افزارهای کند بلکه توسط قابلیت‌های ذهنی خود ما تعیین می‌گردد. تجربه به ما می‌گوید که اغلب برنامه‌ها را می‌توان به نحو قابل ملاحظه‌ای بهبود بخشید، اطمینان‌پذیرتر و اقتصادی‌تر کرد و استفاده از آن‌ها را آسان‌تر ساخت.»[۳۱۱]

چیزی که می‌خواهم بگویم، قانون مارتین رایسر است که خود شما هم به آن اشاره کرده‌اید. می‌گوید:

«نرم‌افزار با آهنگی سریع‌تر از سریع‌تر شدن سخت‌افزار، کندتر می‌شود»[۳۰۶]

و حال به پرسش من دربارهٔ نرم‌افزار مایکروسافت می‌رسیم. آنقدر پیچیده شده است که هیچ فردی به تنهایی نمی‌تواند کلّ سیستم را درک کند. آیا این درست است که برنامه‌نویسان مایکروسافت مجبورند که فقط قابلیت‌های جدیدی به نرم‌افزارشان بیفزایند و در نتیجه، نرم‌افزارها فربه‌تر می‌شوند؟

ویرث: بله.

دِی‌لایت: نرم‌افزار فربه دقیقاً به چه معنی است؟ شما چگونه آن را تشخیص می‌دهید؟

ویرث: مردم به آن «بادکرده» و متورّم نیز می‌گویند. یعنی باد کرده و اندازهٔ بزرگ‌تری پیدا کرده زیرا فرصت نداشته‌اند که بنشینند و دربارهٔ آن فکر کنند و ساختارش را به‌گونه‌ای اصلاح کنند که کوچک‌تر و فشرده‌تر شود.

دِی‌لایت: آیا فکر می‌کنید کسانی که درگیر آن هستند از این مسئله آگاهی دارند؟

ویرث: اوه، بله. ممکن است همه‌شان نه، ولی اغلبشان آگاهی دارند. و مطمئناً در مایکروسافت این آگاهی وجود دارد.

دِی‌لایت: آیا این انگیزه‌ای در آن‌ها به وجود نمی‌آورد که دوباره از نقطهٔ صفر شروع کنند؟

ویرث: بگذارید داستانی برایتان تعریف کنم. من در اینجا در دههٔ ۱۹۸۰ یک دانشجوی دکتری خیلی بااستعداد داشتم که بعداً به آمریکا رفت. او تلاش کرد تا شرکت کوچک خودش را تأسیس کند امّا کارش نگرفت. من به او توصیه کردم که به مایکروسافت برود. او سیستم‌های خانوادهٔ حروف[53] Lilith را طراحی کرد. من در حوالی سال ۱۹۹۴ او را ملاقات کردم و به او گفتم «خوب، ببین تو الان به‌عنوان آدم کاردان و خبره‌ای شناخته می‌شوی که کارهای با ارزشی کرده است. تو الان بر روی سیستم Word کار می‌کنی که خیلی بزرگ و پیچیده شده است. به عقیدهٔ من، اگر دوباره از نقطهٔ صفر شروع کنی، کار عاقلانه‌ای است. چرا این کار را نمی‌کنی؟» او گفت «من این کار را کرده‌ام.» چند سال پیش، او به این ایده رسیده بود. سپس پیشنهادش را به بیل‌گیتس ارائه کرده بود و مطمئن بود که پذیرفته می‌شود. در واقع همین‌طور هم شده بود و گیتس به او گفته بود «باشد، این کار را انجام بده. با گروهت کار را شروع کن.» و آن‌ها سخت مشغول کار شده بودند. نرم‌افزار آن‌ها از قرار معلوم خیلی خوب بوده و سپس مایکروسافت کارمندان بخش فروشش را بین مشتریان فرستاده و پرسیده درباره‌اش چه فکر می‌کنند. همه گفته‌اند «اوه، بله ما از این سیستم نامطمئن Word خسته شده‌ایم. ما خیلی خوشحال می‌شویم که از یک سیستم جدید استفاده کنیم، فقط به یک شرط: باید بتواند تمام متن‌های ما را پردازش کند و باید همساز به بالا[54] باشد.» یعنی در واقع آن‌ها گفته بودند «ما برای یادگیری استفاده از این سیستم موجود، سرمایه‌گذاری زیادی کرده‌ایم و نمی‌خواهیم این سرمایه‌گذاری هدر رود.» در نتیجه، این دانشجوی سابق من گفته بوده «خوب، اگر بخواهیم که کاملاً همساز باشیم، باید مشخصات[55] سیستم Word را در اختیار داشته باشیم. مشخصات آن کجاست؟» پاسخ این بوده که «در خود کد». نتیجه‌اش این شده که کلّ کار کنار گذاشته شده زیرا فقط می‌توانستند سیستم Word را کپی کنند. نتیجهٔ اخلاقی این داستان این است که باید در سرزنش شرکت‌های بزرگ به خاطر این سکون و بی‌حرکتی، کمی محتاط بود. این مشتریان هستند که نمی‌خواهند حرکت کنند و جلو بروند. و این فقط در مورد سیستم Word نیست. برای بسیاری چیزهای دیگر نیز وضع همین‌طور است.

۵-۳ سیستم‌های نهفته[56]

دِی‌لایت: نرم‌افزار فربه، هنگامی که با سیستم‌های نهفته روبرو باشیم (سیستم نهفته، یک سیستم رایانه‌ای است با یک وظیفهٔ مشخص که درون یک سیستم مکانیکی یا الکتریکی بزرگ‌تر کار گذاشته شده باشد. مترجم) می‌تواند به مسائل و مشکلات بزرگ‌تری بینجامد. شما چاک‌مور، خالق زبان برنامه‌نویسی Forth را می‌شناسید؟ او که خبرهٔ سیستم‌های نهفتهٔ بیدرنگ[57] است، با شک و تردید و بدگمانی دربارهٔ لایه‌های نرم‌افزارهای فربه صحبت می‌کند. عبارت زیر، بازتاب افکار اوست:

«رایانه‌ها امروز پر شده‌اند از مقدار زیادی نرم‌افزار که پایگانی[58] از زبان‌ها را در اختیار می‌گذارند. همگذار (اسمبلر) زبانی را برای توصیف برنامهٔ مترجم (کامپایلر) و سیستم عامل تعریف می‌کند. سیستم عامل زبانی را برای کنترل برنامه تعریف می‌کند. برنامهٔ مترجم زبانی را برای برنامه‌های کاربردی تعریف می‌کند. برنامهٔ کاربردی زبانی را برای ورودی‌هایش تعریف می‌کند. کاربر ممکن است تمام این زبان‌ها را بداند یا نداند امّا آن‌ها در آنجا وجود دارند. آن‌ها بین کاربر و رایانه‌اش می‌ایستند و محدودیت‌هایی را بر روی آنچه می‌تواند انجام دهد و هزینه‌ای که برایش خواهد داشت، تحمیل می‌کنند. و هزینه‌ای که دارد به خاطر این است که این پایگان وسیع زبان‌ها مستلزم سرمایه‌گذاری زیادی از نظر وقت انسان و ماشین برای تولید و نیز تلاش مشابهی برای نگهداری آن‌ها می‌باشد. هزینهٔ مستندسازی این برنامه‌ها و خواندن مستندات نیز بسیار زیاد است. و پس از همهٔ این تلاش‌ها، هنوز برنامه‌ها پر از خطا هستند، استفاده از آن‌ها ناراحت‌کننده و ناخوشایند است و هیچکس را راضی نمی‌کند.»

چاک‌مور، به‌عنوان راه‌حل، پیشنهاد می‌کند که این پایگان عظیم زبان‌ها با تنها یک لایه جایگزین گردد. این یک رویکرد افراطی است امّا کاری است که او با سیستم Forth خود کرده است. او در بخش صنعت کار می‌کند، بیشتر به‌عنوان مشاور فناوری‌های نوآورانه تا به‌عنوان فردی که براساس پژوهش‌های بازار عمل می‌کند. یعنی او لازم نیست به مایکروسافت یا دیگران گوش کند، او رئیس خودش است و در واقع بسیار هم موفق است. به نظر می‌رسد که او در سیستم‌های نهفته، جایگاه مناسب استعداد و توانایی‌هایش را یافته است، احتمالاً به خاطر این که رویکرد نرم‌افزار فربه در آنجا کارایی ندارد. به‌طور خلاصه، او توصیه می‌کند که باید از شرّ تمام این لایه‌ها خلاص شد. نه تنها به خاطر این که حاوی خطاهای بسیاری هستند، بلکه وقتی صحبت از سیستم‌های بیدرنگ باشد ...

ویرث: سربار خیلی زیادی هم وجود خواهد داشت.

دِی‌لایت: بله، و تمام این لایه‌های نرم‌افزاری باعث پیش‌بینی‌ناپذیری می‌گردند. وقتی به زبان اُبرون شما نگاه می‌کنیم، در فلسفه‌اش مشابهت‌هایی را می‌بینیم.

ویرث: بله. من فکر می‌کنم برای ساخت سیستم، باید سعی کنیم از یک زبان استفاده کنیم. همان‌طور که نشان داده‌ایم، هیچ دلیلی برای این که این کار را نکنیم وجود ندارد. ما تمام سیستم

53- font
54- upward compatible
55- specification
56- embedded
57- real-time
58- hierarchy

را از گرداننده‌های‌دستگاه[۵۹] تا ویراستارهای گرافیکی و برنامه‌های کاربردی، به یک زبان واحد اُبرون توصیف کرده‌ایم که بسیار ساده‌تر از جاوا و امثال آن است. بنابراین من هم با نظر چاک‌مور در این مورد موافقم. اگر فقط یک زبان داشته باشید، بسیار ساده‌تر هم می‌باشد. به «یکپارچه‌سازی» نیازی نیست. از سوی دیگر، اگر آینده‌ای در طراحی زبان‌های جدیدتر وجود داشته باشد، زبان‌هایی برای محدوده‌های کاربردی خاص است. برای نمونه، یک طراح ماشین باید زبانی در اختیار داشته باشد که ماشین‌هایش را انتزاعی کند. برای کاربردهای خاص، به نظر من نیاز بسیار بیشتری به زبان‌های مناسب‌سازی شده[۶۰] وجود دارد. امّا وقتی صحبت از نرم‌افزار سیستم باشد، من با او موافقم. اُبرون بی‌گمان نوادهٔ غیرمستقیم پاسکال و ماجولا است. همان‌طور که پاسکال ایدهٔ برنامه‌نویسی ساخت‌یافته را در بردارد، ماجولا نیز دربردارندهٔ ایده‌های پودمانگی[۶۱] و ترجمهٔ[۶۲] جداگانه است و اُبرون همهٔ این‌ها را به همراه شیءگرایی[۶۳] دارد. بنابراین، زبان‌ها واقعاً فقط وسیله‌ای برای صورت‌بندی ایده‌ها و پشتیبانی از یک نظام هستند، نه این که به خودی خود هدف باشند.

دِی‌لایت: اُبرون وسیله‌ای برای توسعه‌پذیری نیز هست. شما بارها این را گفته‌اید (**ویرث**: بله) و نیز برای سیستم‌های بیدرنگ.

ویرث: نه به‌طور خاص.

دِی‌لایت: امّا برای آن هم می‌تواند به کار گرفته شود.

ویرث: درست است. مانعی بر سر راه آن نیست. بهتر است این‌طور بگوئیم.

۵-۳-۱ توسعه‌پذیری در مقابل کارایی

دِی‌لایت: برای من خیلی جالب است زیرا همیشه فکر می‌کردم در نظر گرفتن توسعه‌پذیری به نرم‌افزاری فربه و چندلایه می‌انجامد. امّا ظاهراً این‌طور نیست. یعنی می‌شود سیستم‌های توسعه‌پذیر و در عین حال کم حجم و کوچک داشت.

ویرث: بله، همین‌طور است. سیستمی که درگیر سازوکارهای سنگین نباشد.

دِی‌لایت: حتی داشتن یک سیستم بی‌درنگ شیءگرا هم امکان‌پذیر است زیرا شما در یکی از مقاله‌هایتان توضیح داده‌اید که چگونه بالگردهای مدل را هدایت می‌کردید[۳۰۹].

ویرث: بله، هر چند شیءگرایی نقش چندانی در آنجا بازی نمی‌کرد. امّا به عقیدهٔ من، توسعه‌پذیری و شیءگرایی کم و بیش یک چیز هستند. یا شیءگرایی، پیاده‌سازی توسعه پذیری است. شیءگرایی به شما اجازه می‌دهد که ساختمان داده‌های موجود را بدون تغییر آنچه وجود دارد، توسعه دهید. شما می‌توانید برخی از ساختمان داده‌ها را با الحاق ویژگی‌های[۶۴] تازه و الحاق عملگرها[۶۵] به آن‌ها، برای کاربردهای معیّنی اختصاصی کنید. این زیربنای توسعه‌پذیری است.

دِی‌لایت: امّا اگر این توسعه‌پذیری باعث شود که مجبور به مقدار زیادی بررسی‌های زمان اجرا شوید چه؟

ویرث: خوب، نکته همین جاست که طراحی سیستم‌تان را طوی انجام دهید که مجبور به این کار نشوید.

دِی‌لایت: این نکتهٔ اساسی و لُبّ کلام اُبرون است

ویرث: شما اگر بخواهید سیستم مطمئن و ایمنی همراه با انعطاف‌پذیری نوع[۶۶] داشته باشید، نمی‌توانید بررسی‌های زمان اجرا را به صفر برسانید. ایدهٔ من از زمان الگول و پاسکال همیشه این بوده است که یک سیستم نوع ایمن و بررسی نوع ایستا[۶۷] که توسط برنامهٔ مترجم انجام شود داشته باشیم و این کار را گردن سیستم زمان اجرا نیندازیم. اگر سیستم شما شیءگرا باشد و اگر توسعه همواره همساز با نوع پایه‌اش باشد، در این صورت به بررسی‌های زمان اجرا نیاز چندانی نخواهد بود. نکتهٔ اصلی این است که آن را تا حدّ امکان کم‌هزینه کنید. در اُبرون این فقط به یک دستور مقایسه خلاصه شده و بنابراین، سربار ناشی از بررسی، ناچیز و قابل اغماض است.

در مورد زیرروال‌ها[۶۸] هم همین‌طور است. شما سربار کمی برای فراخوانی و برگشت و برای پارامترگذاری نشانی‌دهی غیرمستقیم[۶۹] و امثال آن می‌پردازید. پرسشی که همواره وجود دارد این است: آیا این کار و تلاش بیشتر، ارزشش را دارد؟ اگر این کار به قدر کافی کم و ناچیز باشد، مانند آنچه اکنون برای رویّه‌ها و پارامترگذاری‌ها انجام می‌گیرد، مردم اصلاً درباره‌اش فکر هم نمی‌کنند.

این بررسی نوع مطمئناً مزیّت بزرگی است و کمک زیادی می‌کند. من فکر می‌کنم مهم‌ترین دستاورد پیاده‌سازی اُبرون، همین پیاده‌سازی کارآمد بررسی نوع، حتی برای سیستم‌های توسعه‌پذیر، بوده است. اگر نوع‌سازی ایستا[۷۰] داشته باشید، می‌توانید پس از آن که برنامهٔ مترجم وارد عمل شد، فراموشش کنید. در مورد سیستم‌های توسعه‌پذیر که به بررسی زمان اجرا نیاز است، پردازنده‌های سریع امروزی، سربار زمان اجرای آن را کاملاً قابل اغماض می‌سازند.

ما ناکارآمدی‌های دیگری داریم که هزار بار اثرگذارتر هستند. اگر به سیستم‌های ویندوز یا یونیکس نگاه کنید، یک پودمان[۷۱] یا بسته[۷۲]، صدها بار در بسته‌های کاربردی مختلف حضور دارد (در اُبرون هیچ پودمانی بیش از یکبار نیامده است.) این سیستم‌ها مگابایت‌های زیادی را اشغال می‌کنند، نخست به خاطر این که این مگابایت‌ها در دسترسند و دوم چون به خوبی در موردش فکر نشده است. امّا چون

59- device driver
60- tailored
61- modularisation
62- compilation
63- object orientation

64- attribute
65- operator
66- type
67- static type checking
68- subroutine
69- indirect addressing
70- static typing
71- module
72- package

این مگابایت‌ها امروز به آسانی در دسترس قرار دارند، هیچکس به راه‌حل‌های بهتر فکر نمی‌کند.

۵-۳-۲ شروع از نقطهٔ صفر

دِی‌لایت: به بحث سیستم‌های نهفتهٔ بیدرنگ بازگردیم. شما نیز همانند چاک‌مور، نحوهٔ کارکرد[۷۳] را کنار گذاشته‌اید. شما نوشته‌اید:
«تصمیم بعدی، حذف RT-OS (سیستم عامل بیدرنگ) بود زیرا به نظر می‌رسید اساساً بدون فرایندهای همروند[۷۴] و به شکل ریسه‌ها[۷۵] قابل انجام باشد.»[۳۰۹]

ویرث: شما حالا دارید دربارهٔ پروژهٔ بالگرد صحبت می‌کنید. (خندهٔ **ویرث**) حالا ما بیشتر به زمان حال نزدیک می‌شویم.

دِی‌لایت: و حافظهٔ نهان[۷۶] و ساختن خط لوله[۷۷]. شما آن‌ها را هم کنار گذاشته‌اید.

ویرث: بله. من با آن گروهی که بالگرد مدل را ساخته بودند در تماس بودم زیرا موفقیت آن‌ها در یک مسابقه را شنیده بودم. من می‌خواستم نگاهی به کار آن‌ها بیندازم و ببینم چکار کرده‌اند. آن‌ها به من توضیح دادند که کارشان را با پردازندهٔ ۴۸۶ دوگانه[۷۸] و یک سیستم عامل بیدرنگ انجام داده‌اند. من نگاه دقیق‌تری به آن کردم و متوجه شدم که در زیر یک الگوریتم کوچک، مقدار زیادی تجهیزات و ماشین‌آلات قرار داده‌اند. اوّل از همه، به آن‌ها گفتم که به سیستم عامل بیدرنگ نیازی نیست. بنابراین، از شرّ آن خلاص شدیم. یادتان باشد که آن‌ها مهندسان مکانیک بودند و بر نرم‌افزارهای تجاری اتکاء داشتند. دومین چیزی که از شرّش خلاص شدیم پردازنده‌ها بود. به جای یک جعبهٔ بزرگ، من همه چیز را بر روی یک تخته مدار کوچک به همراه یک پردازندهٔ کم مصرف ARM، قرار دادم. حالا به یک بیستم توان قبلی بیشتر نیاز نبود. یعنی باتری می‌توانست کوچک‌تر و ظریف‌تر باشد و طول مدّت پرواز هم بسیار بیشتر می‌شد. الگوریتم آنقدر ساده بود که اصلاً به سیستم عامل نیاز نداشتم. این در واقع خیلی تعجّب برانگیز بود. اگر شما با دقّت به چیزی نگاه کنید، امکانات ساده‌سازی پدیدار می‌شود. چون ایجاد یا حذف فرایندها در کار نبود و فقط دو فرایند داشتیم، می‌توانستیم همه چیز را در همان‌ها خلاصه کنیم.

دِی‌لایت: از نظر تاریخی، دلیلی که سیستم‌های عامل ارائه شدند این بود که چند کاربر می‌خواستند برنامه‌هایشان را روی یک رایانهٔ گران‌قیمت اجرا کنند. امّا در این موردی که شما گفتید، واقعاً نیازی به سیستم عامل نبود. آیا این مثال دیگری از این واقعیت نیست که مردم کورکورانه راه‌حل‌های گذشته را دوباره به کار می‌گیرند تا آن که فرد باتجربه و خبره‌ای پیدا شود و به آن‌ها بگوید که می‌توانند آن را کنار بگذارند و به آن نیازی ندارند؟ در مورد حافظهٔ نهان نیز که شما آن را کنار گذاشتید، وضع همین‌طور بود. بازبه‌کارگیری کورکورانهٔ قابلیت‌های کارکردی، هزینهٔ خاص خود را دارد. جدا کردن پودمان‌ها از هم می‌تواند به دستیابی‌های کمتر حافظه و در نتیجه، مصرف کمتر انرژی بینجامد. این برای مصرف باتری در تلفن‌های همراه نیز اهمیت دارد.

ویرث: شما برای همهٔ چیزهایی که از وجودشان آگاهی ندارید نیز باید هزینه بپردازید. مهندسی خوب از تولید چیزهای زائد و اضافی پرهیز می‌کند. امّا البته، من مهندسان مکانیک را درک می‌کنم. آن‌ها چیزهایی دربارهٔ زبان‌ها، سیستم‌های عامل و سیستم‌های بیدرنگ شنیده بودند و با خود فکر کرده بودند که این‌ها مسئلهٔ آن‌ها را حل خواهند کرد. آن‌ها باید می‌فهمیدند که نیازی به این‌ها ندارند امّا آنچنان درگیر مسائل سیستم‌های کنترل خود بودند که به سیستم عامل بیدرنگ نگاه نکرده بودند. برای آن‌ها مثل یک جعبهٔ سیاه بود.

دِی‌لایت: هنگامی که دربارهٔ این چیزها با دیگران صحبت می‌کنم، واکنشی که غالباً نشان می‌دهند این است که سیستم‌ها آنقدر پیچیده می‌شوند که دیگر امکان ندارد یک نفر به تنهایی کلّ سیستم را درک کند. امّا اگر دربارهٔ سیستم‌های بیدرنگ مثل بالگرد مدل صحبت کنیم، شما می‌گوئید:
«طراح سیستم‌های نهفته باید مهندس مکانیک، مهندس برق و مهندس نرم‌افزار، همگی با هم باشد.»[۳۰۹]
هنزینگر و سیفاکیس هم حرفی شبیه به این زده‌اند:
«جداسازی محاسبات (نرم‌افزار) از چیزهای فیزیکی (بُن‌سازه[۷۹] و محیط)، برای سیستم‌های نهفته درست عمل نمی‌کند. به جای آن، طراحی سیستم‌های نهفته به رویکرد همه‌جانبه‌تری نیاز دارد تا بُن‌انگاره‌های[۸۰] اساسی از طراحی نرم‌افزار و سخت‌افزار و نظریهٔ کنترل را یکپارچه سازد.»[۱۲۲]

ویرث: بله، شما باید حداقل یک نفر در گروهتان داشته باشید که نگاه کلّ نگری داشته باشد. در غیر این‌صورت، باید خیلی مواظب باشید که چیزهایی بیش از آنچه واقعاً نیاز دارید به دست نیاورید. آن‌ها باید می‌دانستند که سیستم عامل بیدرنگ چقدر فضا به خود اختصاص می‌دهد و چقدر سربار ایجاد می‌کند. یعنی باید مشخصات خدماتی را که توسط سیستم داده می‌شود می‌دانستند و سپس تحلیل می‌کردند که کدام بخش از خدمات واقعاً مورد نیازشان بود. آن‌ها تحلیل دقیقی نکرده بودند.

دِی‌لایت: به بحث سیستم‌های فربه بازگردیم. من مایلم به دایکسترا و کارهایش بر روی سیستم عامل THE اشاره کنم. او آن سیستم را با قراردادن چند لایهٔ برنامه، طراحی کرد. لایه‌های پائینی با پردازنده‌ها و حافظهٔ فیزیکی سروکار داشتند و در نتیجه، لایه‌های بالاتر دیگر کاری با آن‌ها نداشتند. من می‌توانم درک کنم که این انتزاع، برای ساختن یک سیستم عامل همه منظوره خیلی مفید است. امّا اگر با سیستم عامل بیدرنگ سروکار داشته باشیم، که به گفتهٔ متخصصان

73- functionality
74- concurrent processes
75- thread
76- cache
77- pipelining
78- dual
79- platform
80- paradigms

و خبرگان باید از جنبه‌های مکانیکی و فیزیکی سیستم هم آگاهی داشته باشید، آیا امکان ندارد که افراد به‌طور کورکورانه همان رویکرد طراحی دایکسترا را ناآگانه دوباره به کار گیرند؟ (خندهٔ **ویرث**) شما در سیستم بالگرد، مفهوم وقفه[81] را هم کنار گذاشتید. درست است؟

ویرث: من فقط از یک وقفه استفاده کردم و آن هم وقفهٔ ساعتی بود که هر هزارم ثانیه، سازوکاری را به کار می‌انداخت که داده‌های حسگر را جمع‌آوری می‌کرد و داده‌های دستگاه کنترل را با نرخ ثابتی بیرون می‌داد. البته برنامه‌های کاربردی بسیاری هستند که این برای آن‌ها کافی نیست. من به خوبی از این موضوع آگاهی دارم. امّا من بر روی این کاربرد تمرکز کردم. انتزاعی و مجرّدسازی همیشه مفهوم قشنگی است و خیلی کمک می‌کند امّا نباید در مورد آن مبالغه کرد. شما نباید آنقدر از کاربردتان دور شوید که دیگر ندانید کجا هست. من از خودم پرسیدم کدامیک از این همه فرایند بیدرنگ واقعاً مورد نیاز است؟ و به این نتیجه رسیدم که فقط همین ساعت تک‌چرخه‌ای[82] کفایت می‌کند.

۵-۳-۳ اِیدا[83]

دِی‌لایت: یک مثال دیگر، زبان برنامه‌نویسی اِیداست. از بسیاری از دانشگاهیان درخواست شد که آن را مورد بررسی دقیق قرار دهند.

ویرث: از جمله هور (خندهٔ **ویرث**)

دِی‌لایت: شما، هور، دایکسترا و دیگران. البته نه تنها دانشگاهیان. وزارت دفاع آمریکا (DoD) واقعاً تلاش کرد تا نظر بهترین متخصصان وخبرگان جامعهٔ بین‌المللی رایانش را به دست آورد. به این خاطر، طراحی زبان را در نسخه‌های متعدّد برای چند نفر فرستادند امّا نهایتاً دانشگاهیان از آن خوششان نیامد. به گفتهٔ شما:

«اِیدا بر پایهٔ فهرست بلند بالایی از نیازمندی‌ها طراحی شده بود که طراحان نمی‌توانستند آن‌ها را کنترل کنند. این نیازمندی‌ها پیچیده، گسترده، تا حدودی ناسازگار و برخی حتی متناقض بودند. در نتیجه، پیچیدگی آن فراتر از آن گشته بود که برای من پذیرفتنی باشد.»[۳۰۸]

و من این را در نوشته‌های دایکسترا هم یافتم. آیا نمونه‌هایی از آن را به یاد می‌آورید؟ آیا به نظر شما ویژگی‌های خوبی هم در اِیدا وجود داشت یا تماماً بد بود؟

ویرث: مطمئناً ویژگی‌های خوبی هم داشت. آن‌ها کلّ ایدهٔ برنامه‌نویسی ساخت‌یافته و ساختمان داده‌ها را گرفته بودند. در واقع باید بگویم از پاسکال یا ماجولا. تونی هور و من برای یک هفته مشاوره به کالیفرنیا دعوت شدیم. چهار پیشنهاد برای ارسال به وزارت دفاع وجود داشت و ما به مرکز پژوهشی استنفورد (SRI) مشاوره دادیم. ما کار به درد بخوری برای آن‌ها نکردیم زیرا گفتیم: «از شرّ این خود را خلاص کنید، از شرّ آن خود را خلاص کنید، این یکی هم زائد است، ساده‌ترش کنید.» این چیزی نبود که وزارت دفاع می‌خواست. هور یکبار گفت: «ببینید، چیزی که می‌توانم به شما توصیه کنم این است که از پاسکال استفاده کنید، حاضر و آماده است و ما می‌دانیم که چیست.» امّا آن‌ها زبان بزرگ خودشان را می‌خواستند و البته آنچه روی داد هم همان چیزی بود که ما در آن هفتهٔ مشاوره دیدیم: چیز خیلی بزرگی از آب درآمد. همیشه وقتی کمیته‌های مختلف دارید، همین اتفاق می‌افتد زیرا هر کس تلاش می‌کند ایده‌های خود را جا بیندازد. در انتها، آنچه به دست می‌آید، اجتماع[84] ایده‌هاست نه اشتراک[85] آن‌ها. (خندهٔ **ویرث**) اگر فقط یک نگرانی دربارهٔ اِیدا داشته باشم، این است که خیلی بزرگ، خیلی دست و پاگیر و خیلی پیچیده است. همین توصیف‌ها در مورد جاوا و C# نیز صدق می‌کند. آن‌ها زیر وزن‌شان لِه شده‌اند. خیلی متأسفم که این داستان دوباره تکرار شده است. در مورد پی‌ال‌وان هم همین وضع بود. ما همگی آن را در اواسط دههٔ ۱۹۶۰ تجربه کرده بودیم ولی همان اتفاق باز هم دربارهٔ الگول ۶۸ افتاد.

۵-۴ پسگفتار: وضعیت اندوهبار

دِی‌لایت: هنگامی که کارهای شما را در کنار کارهای هور و دایکسترا مطالعه می‌کنم، به خوبی متوجه می‌شوم که شما برای خردورزی و برای عقل سلیم[86] فریاد می‌زدید. (**ویرث**: بله) من کنجکاوم که بدانم امروز چند نفر از دانشگاهیان هنوز این فریاد را می‌شنوند. یا آن که آن‌ها نیز تحت تلقین پژوهش‌های از نوع مایکروسافت و گوگل قرار گرفته‌اند؟

ویرث: متأسفم که این‌گونه است. بله، من هم کمی ناامیدم. و گاهی اوقات افسرده. من از مرگ دایکسترا خیلی متأسف شدم. بلافاصله متوجه شدم که یک صدای مهم خاموش شد. و هیچ صدایی هم جایگزین آن نشده است. برخی افراد می‌گفتند دیگر کسی او را جدّی نمی‌گیرد، امّا هنوز وجود آن صدا اهمیت داشت. او در فرمول‌بندی ایده‌ها استعداد درخشانی داشت.

دِی‌لایت: به جای موشکافی زبان‌های برنامه‌نویسی معروف، می‌توانیم دربارهٔ زبان‌های طراحی سخت‌افزار مانند وریلوگ[87] و VHDL هم صحبت کنیم.

ویرث: اگر به دقّت به آن‌ها نگاه کنید، وریلوگ نشانگر شیوه‌ای است که آمریکایی‌ها طراحی سخت‌افزار را به زبان C انجام می‌دهند. وریلوگ همان قواعد نحوی C را اقتباس کرده و VHDL نیز قواعد نحوی اِیدا را. من واقعاً نمی‌خواهم از این دو زبان انتقاد کنم یا آن‌ها را با هم مقایسه کنم، امّا نکته‌ای که وجود دارد این است که هر دو سیستم خیلی چیزهای حجیم و سنگینی شده‌اند. البته ترجمهٔ یک برنامهٔ نرم‌افزاری و ترجمهٔ یک برنامهٔ سخت‌افزاری به درون یک مدار، دو چیز کاملاً متفاوت است.

81- interrupt
82- one-cycle clock
83- Ada
84- union
85- intersection
86- common sense
87- Verilog

دومی بسیار پیچیده‌تر است زیرا باید طرح‌بندی‌ها[88] را تولید کنید و بهینه‌سازی‌های[89] زیادی انجام دهید. با وجود این، من حس می‌کنم دوباره به دهۀ ۱۹۶۰ برگشته‌ایم. برای مثال، هنگامی که من برخی اصلاحات بر روی برنامه یا مدارم انجام می‌دهم و سپس دکمۀ ترجمه (کامپایل) را فشار می‌دهم، باید نیمساعت صبر کنم تا دوباره ترجمه شود! و این برای برنامه‌ای است که فقط چهار صفحه است. در این فاصله من باید قهوه بنوشم یا سرم را به کار دیگری گرم کنم. به‌علاوه، چندین بار تاکنون اتفاق افتاده است که با وجودی که ترجمه به طرز موفقیت‌آمیزی خاتمه یافته، هنوز متوجه شده‌ام که یک چیزی درست اجرا نمی‌شود. با آزمایش‌های بیشتر، خطای برنامه‌ام را پیدا کردم، خطایی که به راحتی با یک بررسی نوع[90] ساده قابل تشخیص بود. یکبار دیگر که فراموش کرده بودم بُعد صحیح یک متغیر را مشخص کنم، به من هشدار[91] نداد. در عوض ۲۰۰ هشدار دیگر داد- آنقدر زیاد که آدم آن‌ها را نادیده می‌گیرد. یا یکبار فراموش کرده بودم متغیری را اعلان[92] کنم و باز هم اشتباهی بود که خودم مجبور شدم کشف کنم. متغیر باید به یک آرایۀ ۱۶ بیتی مربوط می‌شد امّا سیستم آن را در یک بیت قرار داده بود. سیستم به من نگفت «اعلان را فراموش کرده‌ای» بلکه در عوض، اعلان خودش را جایگزین کرده بود. این مربوط به سال ۲۰۱۰ بود. باور نکردنی است. غالباً به نظر می‌رسد این افراد آنچه را دانشمندان رایانه ظرف ۴۰ سال گذشته تجربه کرده‌اند کاملاً نادیده گرفته‌اند.

دِی‌لایت: من فکر نمی‌کنم آن‌ها به آثار مکتوب نگاه کرده باشند زیرا باز همگی خیلی سرشان شلوغ است. وقتی به آن‌ها گزارش فنی حاوی یک راه‌حل را نشان می‌دهید، می‌گویند «اوه، این‌ها فقط ریاضیات است.»

ویرث: آن‌ها همیشه بهانه‌ای پیدا می‌کنند.

دِی‌لایت: مقالات زبان‌های برنامه‌نویسی که در همایش‌های سطح بالا ارائه شده‌اند، خیلی صوری هستند. امّا تکنیک‌ها باید در C یا C++ کار کنند. کمی خنده‌دار است زیرا شما به‌عنوان پژوهشگر باید رو به عقب کار کنید: با یک زبان بد آغاز کنید و سپس سعی کنید با آن چیز خوبی بسازید.

ویرث: و همسازی به بالا[93] را هم حفظ کنید، یعنی سیستم بازبینی نوع ممکن است خطاها را تشخیص دهد یا ندهد.

دِی‌لایت: متأسفانه این وضعیت گریبانگیر نرم‌افزار شده است.

ویرث: بله، همین‌طور است.

دِی‌لایت: نیکلاوس **ویرث**، بابت این مصاحبه از شما تشکر می‌کنم.

ویرث: خواهش می‌کنم!

88- layout
89- optimization
90- type check
91- warning
92- declare

93- upward compatibility

بهترین هدیهٔ نوروزی برای کارکنان بخش فناوری اطلاعات کشور

جدید

بازنگری در کار

نویسنده: جیسون فرید و دیوید هاین مایرهنسون

مترجم: ابراهیم نقیب زاده مشایخ

قیمت: ۱۰۰۰۰ تومان

سال انتشار: ۱۳۹۵

(تعداد محدود)

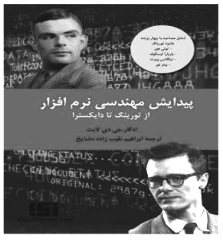
چگونه نهضت ضد فرهنگ دهه شصت به شکل گیری صنعت رایانه های شخصی انجامید

نویسنده: جان مارکف

مترجم: ابراهیم نقیب‌زاده مشایخ

قیمت: ۶۰۰۰ تومان

سال انتشار: ۱۳۸۹



پیدایش مهندسی نرم افزار

نویسنده :ادگار.دی. جی لایت

مترجم : ابراهیم نقیب‌زاده مشایخ

قیمت : ۱۵۰۰۰ تومان

سال انتشار: ۱۳۹۴

مصاحبه با ۱۵ تن از برجسته ترین برنامه نویسان جهان

نویسنده: پیتر سیبل

مترجم: ابراهیم نقیب زاده مشایخ

قیمت: ۲۰۰۰۰

سال انتشار:۱۳۹۱

چگونه اینترنت داوود را به جالوت تازه ای بدل ساخت

نویسنده :نیکو میل

مترجم: ابراهیم نقیب‌زاده مشایخ

قیمت: ۱۰۰۰ تومان

سال انتشار:۱۳۹۳

بزرگان دانش رایانه

نویسنده : دنیس الیوت شاشا و کتی لارز

مترجم: ابراهیم نقیب‌زاده مشایخ

قیمت:۲۵۰۰ تومان

سال انتشار:۱۳۸۴

جهت سفارش و دریافت کتاب با دفتر انجمن انفورماتیک‌ایران ۶۶۴۱۲۹۷۶ تماس بگیرید

سفارش‌های بیش از ۱۰جلد کتاب مشمول ۱۵٪ تخفیف می‌گردد

ارائهٔ یک روش ترکیبی مبتنی بر انتشار گوینده ـ شنونده و بلاندل برای تشخیص جوامع به صورت پویا در شبکه‌های اجتماعی*

محمد ستاری
دانشجوی دکتری، دانشکده مهندسی کامپیوتر، دانشگاه اصفهان
پست الکترونیکی: Mohamamdsattari9220@eng.ui.ac.ir

کامران زمانی‌فر
دانشیار دانشکده مهندسی کامپیوتر، دانشگاه اصفهان
پست الکترونیکی: Zamanifar@eng.ui.ac.ir

چکیده

تشخیص جوامع به صورت پویا یکی از مباحث مهم تحقیقاتی در شبکه‌های اجتماعی است. رویکردهای گوناگونی برای تشخیص جوامع به صورت پویا وجود دارد. یکی از پرکاربردترین رویکردها در این زمینه، رویکرد مبتنی بر انتشار برچسب است. روش‌های گوناگونی برای تشخیص جوامع مبتنی بر انتشار برچسب وجود دارد. یکی از این روش‌ها، روش انتشار گوینده – شنونده پویا است.

روش انتشار گوینده-شنونده پویا سرعت بالایی برای تشخیص جوامع به صورت پویا در شبکه‌های اجتماعی دارد. با این وجود، این روش در برخی مواقع در تشخیص صحیح جوامع دچار مشکل خواهد شد. یکی از این مشکلات، این است که یک گره به یک جامعه می‌پیوندد بدون آن‌که اتصالات درونی آن جامعه و تعداد ارتباطات گره با آن جامعه در نظر گرفته شود. این مشکل صحت تشخیص جوامع را در حالت پویا کاهش خواهد داد. در این مقاله، یک روش ترکیبی براساس روش انتشار گوینده-شنونده و روش حریصانهٔ بلاندل ارائه شده است. در ادامه روش ارائه‌شده و روش انتشار گوینده – شنونده پویا بر روی مجموعهٔ داده‌ای دی‌بی‌ال‌پی* پیاده‌سازی شده است. نتایج این پیاده‌سازی نشان می‌دهد که روش ارائه‌شده از لحاظ صحت نسبت به روش انتشار گوینده – شنونده پویا بهتر عمل کرده است. همچنین زمان اجرای دو الگوریتم تفاوت چندانی ندارد.

کلمات کلیدی: روش انتشار گوینده – شنونده پویا، روش بلاندل***، رویکرد مبتنی بر انتشار برچسب تشخیص جوامع به صورت پویا، شبکه‌های اجتماعی

۱- مقدمه

در سال‌های اخیر، شبکه‌های اجتماعی از قبیل فیسبوک و توییتر رشد سریع و قابل توجهی داشته‌اند به طوری که این شبکه‌ها توانسته‌اند میلیاردها کاربر را از سراسر جهان به‌سوی خود جلب کنند. به‌عنوان مثال شبکهٔ فیسبوک بیش از یک میلیارد کاربر (گره) دارد. یکی از حوزه‌های مهم در شبکه‌های اجتماعی، تشخیص جوامع است.

* این مقاله در بیست و یکمین کنفرانس ملی سالانهٔ انجمن کامپیوتر ایران، ۱۸-۲۰ اسفند ۹۴، ارائه شده است

** DBLP
*** Blondel

تشـــخیص جوامع را می‌توان متناظر با خوشـــه‌بندی گره‌های موجود در شـــبکه، براساس توپولوژی شبکه در نظر گرفت، در حالی که هیچ اطلاعاتی در مورد اندازهٔ جوامع و ســـاختار ارتباطی آن‌ها وجود ندارد [۲].

تشـــخیص جوامع، کاربردهای متعـــددی دارد. به‌عنوان مثال، جهت ارسال یک پیام به تعدادی از افراد در یک شبکه اجتماعی، در صورتی که جوامعی که این افراد عضوی از آن‌ها هســـتند، شـــناخته شـــده باشـــند، سبب می‌شود که از ارســـال پیام‌های اضافی جلوگیری شود. بدین‌صورت که برای افرادی که عضو یک جامعه هســـتند، کافیست پیغام یک بار به شناسه جامعه فرستاده شود[۳]. تشخیص جوامع به دو صورت ایستا و پویا انجام می‌شود.

بـــا توجه به ذات پویایی شـــبکه‌های اجتماعی، تشـــخیص جامعه به صورت پویا در ســـال‌های اخیر بیشتر مورد توجه قرار گرفته است. در تشخیص جوامع به صورت پویا، شـــبکهٔ اجتماعی به نماهای مختلف تقسیم می‌شوند. هر یک از این نماها نشان‌دهندهٔ گره‌ها و روابط آن‌ها در دوره‌های زمانی مختلف است. رویکردهای مختلفی برای تشخیص جامعه به صورت پویا وجود دارند. رویکرد اول، رویکرد مســـتقل است که در آن برای هر نما به صورت جداگانه، الگوریتم تشـــخیص بر روی کل گراف اعمال می‌شـــود. در واقع از اطلاعات نمای قبلی اســـتفاده نمی‌شـــود. با توجه به این‌که این رویکرد مدنظر این تحقیق نیست از ذکر جزییات آن خودداری شده است.

رویکرد دوم، رویکرد تکاملی اســـت. در این رویکرد از اطلاعات نمای قبلی در نمای جدید اســـتفاده می‌شـــود. در نمای اولیـــه، الگوریتم تشـــخیص بر روی کل گراف اعمـــال می‌شـــود و در نماهای بعدی، الگوریتم تشـــخیص بر روی تغییرات رخ داده در بین دو نما و جوامع تشخیص‌داده‌شده در نمای قبلی پیاده‌سازی می‌شود. پس ورودی هر نما، جوامع تشخیص‌داده‌شـــده در نمای قبلـــی و تغییرات رخ‌داده در نمای فعلی نســـبت به نمای قبلی اســـت. همچنین خروجی هر نما، جوامع تشخیص‌داده‌شده در نمای فعلی است.

روش‌های مختلفی برای تشخیص تکاملی جوامع به صورت پویا وجود دارد. برخی از این روش‌ها از ایدهٔ نظریهٔ بازی استفاده کرده‌اند [۱]. در این روش‌ها گره‌ها به‌عنوان یک عامل در نظر گرفته شـــده که در یک بازی چندنفره به دنبال بهبود بهره‌وری خود هســـتند. در اینجا بهبود بهره‌وری را می‌توان متناظر با حضور در یک جامعهٔ بهتر درنظرگرفت. برخی دیگر از آن‌ها از ایده حریصانه بهره برده اند [۲] بدین‌صورت که به صورت حریصانه به دنبال بهبود صحت تشخیص جوامع بوده‌اند. در واقع یک گره در صورتی به یک جامعه می‌پیوندد که حضورش باعث بهبود صحت تشخیص جوامع شود.

برخی دیگر مبتنی بر انتشـــار برچسب هستند. در این روش‌ها، به هر گره یک برچســـب یا مجموعه‌ای از برچسب‌ها اختصاص داده می‌شود که این گره برچسب خودش را با بقیهٔ گره‌ها به اشتراک می‌گذارد. در نهایت برچسب‌های مشابه، نشان‌دهندهٔ جوامع مشابه هستند. بنابراین گره‌هایی که برچسب مشـــابه دارند در یک جامعه قرار می‌گیرند. در بین این دســـته‌ها، روش‌های مبتنی بر انتشـــار برچسب طبق [۴و۵] نســـبت به بقیه بهتر عمل کرده‌اند. از مزایای این دســـته می‌توان به سادگی پیاده‌سازی، سرعت بالای تشـــخیص جوامع و قابلیت اجرای موازی اشاره کرد.

روش‌های مختلفی بر مبنای انتشار برچسب برای تشخیص جوامع به صورت پویا وجود دارد. یکی از مهمترین روش‌های مبتنی بر انتشـــار برچســـب برای تشخیص جامعه به صورت پویا، روش انتشار گوینده – شنونده پویا[1] اســـت [۴]. در این روش، هر گره شامل یک مجموعه برچسب است که با برچسب‌های گره‌های دیگر به اشتراک می‌گذارند. در نهایت برچسب‌های مشابه نشان‌دهندهٔ جوامع مشابه خواهند بود. در این روش در اجراهای مختلف، نتایج متفاوتی حاصل می‌شـــود. با وجود مزایای متعدد، این روش با مشـــکلاتی مواجه است. یکی از این مشـــکلات، این است که در هنگام پیوســـتن یک گره به یک جامعه، تعداد اتصالات درونی آن جامعـــه و تعداد ارتباطات گره با آن جامعه در نظر گرفته نمی‌شـــود. این مشکل باعث کاهش صحت تشخیص جوامع خواهد شد.

۱-۱- کارهای مرتبط

بحث تحلیل شبکه‌های اجتماعی از سال ۱۹۳۰ مطرح شد و به تدریج به اهمیت آن در علم جامعه‌شناسی اضافه شد. یکی از بحث‌های مهم در تحلیل شبکه‌های اجتماعی، بحث تشخیص جوامع است. تشخیص جوامع اولین بار در[۶] مطرح شد و در سال ۲۰۰۲ توسط گیروان و نیومن اولین الگوریتم برای تشخیص جامعه ارائه شد[۷].

تشخیص جامعه در ابتدا در حالت ایستا مطرح شد. بدین‌صورت که یک نما از سیســـتم گرفته شده، سپس جوامع براساس گره‌ها و ارتباطات موجود در این نما، تشـــخیص داده می‌شوند. در ادامه حالت پویا برای تشـــخیص جوامع مطرح شـــد. بدین‌صورت که چندین نما از سیستم گرفته شده و براساس اطلاعات چندین نما، جوامع استخراج می‌شوند. در واقع در این حالت تغییرات در نظر گرفته می‌شـــوند و همچنین از اطلاعات جوامعی که در نماهای قبلی تشـــخیص داده شـــده‌اند، بهره برده می‌شـــود. پرکاربردترین رویکرد موجود برای تشـــخیص جوامع به صورت پویا، رویکرد مبتنی بر انتشـــار برچســـب است. روش‌های مختلفی در ایـــن رویکرد وجود دارد. یکی از مهم‌ترین روش‌ها، روش انتشار گوینده-شنونده پویا است. این مقاله به دنبال بهبود این روش است. برخی دیگر از روش‌های مرتبط در ادامه آمده است.

ژیوری و همکاران [۵] روش رتبه‌بندی برچســـب مبتنی بر زمان[2] را برای تشـــخیص جوامع در شـــبکه‌های اجتماعی ارائه دادند. در این روش در نمای اول، روش رتبه‌بندی برچســـب [3] [۸] اعمال می‌شود. بدین‌صورت که هر گره یک بردار توزیع برچسب به طول n دارد که هر

1- SLPAD
2- LabelRank-T
3- LabelRank

درایهٔ این بردار، احتمال عضویت برچسب متناظر با آن درایه را برای گره موردنظر مشخص می‌کند. در ادامه برای کلیهٔ گره‌ها، بردار توزیع برچسب در طی ۴ گام انتشار،تورم[۴]، بریدن[۵] و بروزرسانی شرطی دوباره مقداردهی خواهد شد. این گام‌ها بین ۵ تا ۲۰ مرتبه تکرار خواهد شد. در نهایت گره‌هایی که برچسب‌های با بیشترین احتمال عضویت‌شان مشابه است، در یک جامعه قرار می‌گیرند.

در نماهای بعدی، گام‌های فوق‌الذکر فقط بر روی گره‌ها و یال‌های تغییر یافته انجام خواهد شد. به طوری که بردار توزیع برچسب برای گره‌های تغییرنیافته تغییر نخواهد کرد، ولی بردار توزیع برچسب برای گره‌های تغییریافته براساس گام‌های ذکرشده در نمای اولیه دوباره مقداردهی خواهد شد. در نهایت مشابه نمای اولیه، مشابهت در برچسب‌ها با بیشترین احتمال عضویت نشان‌دهندهٔ جوامع مشابه است.

لیو و همکاران [۹] روش انتشار برچسب برجستهٔ گسترش‌یافته[۶] که ارتقایافتهٔ روش انتشار برچسب برجسته [۱۰] بود، برای تشخیص جوامع در شبکه‌های اجتماعی در حالت پویا ارائه دادند. در این روش، مفهومی به نام برچسب برجسته برای هر گره تعریف شده است که نشان‌دهندهٔ جامعهٔ اصلی است که گره به آن تعلق دارد. نکته‌ای دیگری که این روش را از روش‌های قبلی مجزا می‌کند، مفهومی به نام واریانس اعتماد است که برای هر گره استفاده می‌شود و نشان‌دهندهٔ میزان اعتماد گره‌های همسایهٔ یک گره به آن گره است. هر چه میزان این واریانس بیشتر باشد، احتمال این‌که یک گره جامعه‌ای مشابه گره‌های همسایه‌اش انتخاب کند، بیشتر خواهد شد. همچنین مفهوم دیگری که در این روش استفاده شده، مفهوم ضریب تعلق است که نشان‌دهندهٔ میزان تعلق یک گره به یک جامعه است. در ابتدا در نمای اولیه، گره‌ها براساس واریانس اعتماد مرتب می‌شوند. سپس به ترتیب برای هر گره، ضریب تعلق گره به برچسب‌هایش محاسبه می‌شود. در نهایت، برای هر گره، برچسبی که بیشترین ضریب تعلق را دارد، نشان‌دهندهٔ جامعهٔ اصلی گره و بقیهٔ برچسب‌ها، نشان‌دهندهٔ جوامع فرعی گره هستند. در نماهای دوم به بعد، روند نمای اولیه تکرار خواهد شد و مشابه نمای اولیه برای هر گره، جامعهٔ اصلی متناظر با برچسب با بیشترین ضریب تعلق خواهد بود و جامعهٔ فرعی متناظر با بقیهٔ برچسب‌ها خواهد بود.

۲- روش انتشار گوینده - شنونده پویا

استون و همکاران [۶] روش انتشار گوینده-شنونده پویا را که توسعه یافتهٔ روش انتشار گوینده-شنونده[۷] [۱۱] است، برای تشخیص جوامع در شبکه‌های اجتماعی در حالت پویا ارائه دادند. در این روش در نمای اولیه به هر گره یک برچسب تخصیص داده می‌شود، به صورتی که برچسب هیچ دو گرهی مشابه نباشد. در ادامه گام‌های ذیل انجام می‌شوند:

۱) یک گره به صورت تصادفی انتخاب می‌شود.

۲) هر کدام از همسایه‌های گره انتخاب شده، پرتکرارترین برچسب یا یک برچسب به صورت تصادفی از بین مجموعهٔ برچسب‌هایشان به گره منتخب ارسال می‌کنند.

۳) گره منتخب از میان برچسب‌های ارسال‌شده از سوی همسایگانش، پرتکرارترین آن‌ها را انتخاب می‌کند.

گام‌های ۱ تا ۳ چندین مرتبه تکرار می‌شود. در ادامه، برای هر گره، برچسبی که کمترین تکرار را دارد، حذف خواهد شد. در آخر، گره‌هایی که برچسب‌های مشابه دارند در یک جامعه قرار خواهند گرفت. در نماهای دوم به بعد، در ابتدا تفاضل دو گراف در دو نمای متوالی به صورت اجتماع یال‌هایی که در یک گراف هستند و در دیگری نیستند، محاسبه شده و سپس راس‌هایی که یکی از دو سر این یال‌ها هستند، راس‌های تغییریافته در نظر گرفته می‌شوند. در ادامه برچسب راس‌های تغییریافته‌ای که در گراف جدید نیستند، حذف می‌شوند.

سپس برای راس‌های تغییریافتهٔ باقیمانده، گام‌های ۱ تا ۴ انجام می‌شود. در نهایت، هر گره برچسبی که بیشترین تکرار را در بین همسایه‌ها داشته باشد، در صورتی که عضو مجموعه برچسب‌هایش نباشد، به مجموعهٔ برچسب‌های خود اضافه می‌کند. این قسمت به‌عنوان ترمیم جامعه است و با این هدف است که جوامع استحکام بهتری یابند. این روش نقاط ضعف مختلفی دارد. یکی از این مشکلات، در نظر نگرفتن تعداد اتصالات درونی آن جامعه و تعداد ارتباطات گره با آن جامعه در هنگام پیوستن یک گره به یک جامعه است. در واقع در برخی مواقع یک گره به جامعه‌ای می‌پیوندد که پتانسیل حضور در آن جامعه را ندارد. این مشکل باعث تشخیص نادرست جوامع برای برخی گره‌ها در حالت پویا خواهد شد. بنابراین دقت روش تشخیص جوامع انتشار گوینده- شنونده را کاهش خواهد داد. در ادامه راه‌حل پیشنهادی این مقاله برای حل مشکل ذکرشده بررسی خواهد شد.

۳- روش پیشنهادی

ایدهٔ رفع مشکل ذکرشده برای روش انتشار گوینده-شنونده پویا، استفاده از روش‌های ترکیبی است. در واقع روش‌های تشخیص جامعه در حالت پویا را می‌توان به دو بخش تقسیم کرد. بخش اول، بخش ایستا است. در این بخش، الگوریتم بر روی کل گراف شبکهٔ اجتماعی اعمال می‌شود. ورودی این بخش، گراف نمای اولیه و خروجی این بخش، جوامع تشخیص‌داده‌شده در نمای اولیه خواهد بود.

بخش دوم، بخش پویا است. در واقع بخش پویا متناظر با نمای دوم به بعد است. در این بخش، الگوریتم بر روی تغییرات مرتبط با

4- inflation
5- cut
6- DLPAE
7- SLPA

گره‌ها و یال‌ها اعمال می‌شـود. این تغییرات می‌تواند شـامل افزودن گره‌ها، افزودن یال‌ها، حذف گره‌ها و حذف یال‌ها باشـد. ورودی این بخش جوامع تشخیص‌داده‌شـده در نمای قبلی و تغییرات رخ‌داده در نمای فعلی نسـبت به نمای قبلی است و خروجی این بخش، جوامع تشخیص داده‌شده در نمای فعلی است.

ایدهٔ ترکیب روش‌هـا را می‌توان متناظر بـا درنظرگرفتن روش‌های متفـاوت برای هر کدام از بخش‌هـای فوق‌الذکر در نظر گرفت. روش پیشنهادی برای بخش ایستا، روش انتشار گوینده-شنونده است. روش پیشنهادی برای بخش پویا، روش بلاندل است. لذا روش پیشنهادی، ترکیبی از دو روش انتشار گوینده - شنونده و روش بلاندل است.

در واقع در نمای اولیه، روش انتشـار گوینده - شنونده [۱۱] بر روی کل گراف شـبکهٔ اجتماعی اجرا خواهد شـد و خروجی این روش که همان جوامع تشخیص‌داده‌شـده است به‌عنوان ورودی به نمای بعدی داده خواهد شـد. در نماهای بعـدی، روش حریصانهٔ بلاندل [۱۲] بر روی جوامـع تشخیص‌داده‌شـده در نمای قبلـی و تغییرات رخ‌داده مرتبـط با گره‌ها و یال‌ها در نمای فعلی نسـبت به نمای قبلی اعمال خواهد شد و خروجی این روش، جوامع تشخیص‌داده‌شده در نماهای دوم به بعد اسـت. در ادامه هر دو بخش روش ترکیبی ارائه‌شـده به صورت جداگانه توضیح داده خواهد شد.

۳-۱- روش انتشار گوینده - شنونده

در نمای اولیه، روش انتشـار گوینده - شنونده اجرا می‌شود. ورودی این روش، کل گراف شـبکهٔ اجتماعی در نمای اولیه شـامل گره‌ها و ارتباطات بین آن‌ها است و خروجی این روش، جوامع تشخیص‌داده‌شده است. این روش بدین صورت عمل می‌کند که به هر گره، یک برچسب به صورت یکتا اعطا می‌شـود. سـپس یکی از ایـن گره‌ها به صورت تصادفی انتخاب می‌شـود. در ادامه همسایه‌های گره منتخب یکی از برچسب‌هایشـان را برای آن گره ارسال می‌کنند. سپس گره منتخب، پرتکرارترین برچسب را در میان برچسب‌های ارسال‌شده برمی‌گزیند و آن را به‌عنوان برچسـب جدید خود خواهد پذیرفت. این روند برای کلیهٔ گره‌ها انجام خواهد شد. پس از اتمام روند برای کلیهٔ گره‌ها، یک برچسب به مجموعه برچسـب‌های گره‌هایی که حداقل یک همسایه دارند، اضافه خواهد شد.

در ادامـه روند ذکرشـده چندین مرتبهٔ دیگر انجام خواهد شـد. در نهایت برای هر گره، برچسـب‌هایی که تعداد تکرارشان کمتر از یک مقدار آسـتانه است، از مجموعه برچسب‌های آن گره حذف می‌شوند و برچسـب‌های باقیمانده، نشـان‌دهندهٔ جوامع خواهند بود. در واقع گره‌هایی که برچسـب‌های مشابه دارند در یک جامعه قرار می‌گیرند. جوامع تشخیص‌داده‌شده توسـط این روش به‌عنوان ورودی به نمای بعدی داده خواهند شد.

۳-۲ روش بلاندل

پـس از آن‌که جوامع در نمای اولیه تشـخیص داده شـد، به‌عنوان ورودی به نمای دوم فرسـتاده می‌شـوند. ورودی دیگـر نمای دوم، تغییرات رخ داده در نمای دوم نسـبت به نمای اول اسـت. سـپس روش بلاندل بر روی ورودی‌های ذکرشـده اعمال می‌شود به صورتی که یک گره در صورتی بـه یک جامعه می‌پیوندد که حضورش باعث بهبود معیار پودمانگی[8] شـود. معیار پودمانگی، مشخص‌کنندهٔ میزان تفکیک‌پذیری گراف شبکه به جوامع است. هر چه گره‌هایی که عضو یک جامعه باشـند، ارتباط بیشتر و گره‌هایی که عضو جوامع متفاوتی هستند، ارتباط کمتری داشته باشـند، پودمانگی بیشتر خواهد شد. روش بلاندل با کمی تغییر اعمال می‌شـود که در ادامه توضیح داده خواهد شد.

این روش بدین‌صورت اعمال می‌شـود که در نماهای دوم به بعد یک گراف جدید ایجاد می‌شـود که در آن هر یک از گره‌های که در نمای جاری به گراف شـبکهٔ اجتماعی اضافه شـده‌اند به‌عنوان یک گره در گراف جدید در نظر گرفته می‌شود. همچنین هر یک از جوامع موجود به‌عنوان گره‌های دیگر این گراف در نظر گرفته خواهد شـد. سـپس یکی از گره‌ها در گراف جدید به صورت تصادفی انتخاب می‌شـود. در ادامه برای گره موردنظر، ارتباطات آن گره با گره‌های مختلف براساس گراف جدید بررسـی خواهد شد. سـپس آن گره با یکی از گره‌های همسـایه‌اش باعث بیشـترین افزایش در معیار پودمانگی شود، ادغام می‌شـوند و یک جامعهٔ جدید را می‌سـازند. گام‌های ذکر شده برای بقیهٔ گره‌ها انجام خواهد شـد. در نماهای سـوم به بعد، همین روند ادامه خواهند یافت تا زمانی که کلیهٔ نماها در روند تشـخیص جامعه در نظر گرفته شوند.

۴- ارزیابی

یکی از چالش‌های تشـخیص جوامع، کمبـود جوامع حقیقی جهت ارزیابی اسـت. در واقع تعداد مجموعه‌های داده‌ای که جوامع حقیقی مشـخص دارند، در حالت پویا بسـیار کم اسـت. یکی از راهکارهای پیشـنهادی برای حل این مشـکل، ایجاد چهارچوب‌هایی است که بتواند رفتار جوامع حقیقی را تقلید کند. در این تحقیق از چهارچوبی که در [۱۳] به‌کار برده شده، استفاده شده است.

برای ایجاد مجموعهٔ داده‌ای سـاختگی، بخشـی از مجموعهٔ داده‌ای به‌عنوان مجموعـهٔ اولیه و بقیهٔ آن به‌عنوان مجموعهٔ تغییرات در نظر گرفته خواهد شـد. سـپس نماها براسـاس این دو مجموعه ساخته می‌شـوند. نمای اولیه متناظر با مجموعهٔ اولیه و نماهای دوم به بعد متناظر با مجموعهٔ تغییرات در نظر گرفته شـده‌اند. با استفاده از این چهارچوب می‌توان از مجموعه‌های داده‌ای ایستا با جوامع حقیقی، در حالت پویا استفاده کرد.

به‌همیـن منظور در این تحقیق از مجموعهٔ داده‌ای دی‌بی‌ال‌پی [۱۴] که یک مجموعهٔ داده‌ای ایسـتا دارای جوامع حقیقی مشخص است،

8- modularity

جدول ۱: نماها و تعداد گره‌ها و یال‌های مجموعهٔ داده‌ای

نما	۱	۲	۳	۴	۵
تعداد گره‌ها	۱۴۳۵۰	۱۵۱۰۰	۱۵۵۵۰	۱۶۵۵۰	۱۶۹۰۰
تعداد یال‌ها	۱۳۹۸۶	۱۴۷۲۴	۱۵۵۳۴	۱۷۲۲۱	۱۷۸۳۴
نما	۶	۷	۸	۹	۱۰
تعداد گره‌ها	۱۷۴۰۰	۱۸۱۲۳	۱۹۰۰۰	۱۹۴۵۰	۲۰۰۰۰
تعداد یال‌ها	۱۸۳۴۵	۱۹۱۲۴	۱۹۷۷۰	۲۰۴۴۵	۲۱۰۲۰

استفاده شـــده است. این مجموعه دارای ۳۱۷۰۸۰ گره و ۱۰۴۹۸۶۶ یال اســـت. از بین ایـــن گره‌ها، حدود ۲۰۰۰۰ تـــا از آن‌ها به صورت تصادفی انتخاب شده‌اند. ســـپس مجموعهٔ منتخب به ۱۰ نما تقسیم شـــده است. در هر نما، تعداد گره‌ها و یال‌ها متفاوت است. روند ایجاد این مجموعهٔ داده‌ای ساختگی در جدول (۱) مشخص شده است.

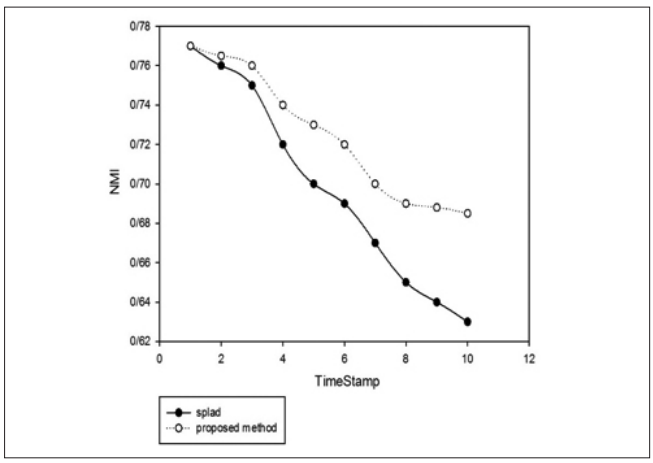
همان‌طور در جدول (۱) نشـــان داده شده است، تعداد گره‌ها و تعداد یال‌ها از یک نما نســـبت به نمای دیگر افزایش پیدا کرده اســـت که این میزان افزایـــش در نماهای مختلف متفاوت اســـت. در مجموعهٔ دی‌بی‌ال‌پی، گره‌ها مرتبط با افراد است. دو فرد در صورتی با هم اتصال دارند که حداقل یک مقالهٔ مشترک با هم داشته باشند. بنابراین تنها رویدادی که برای گره‌ها و یال‌ها اتفاق می‌افتد، رویداد افزودن اســـت و رویداد حذف رخ نمی‌دهد. لذا براســـاس جدول (۱) می‌توان تعداد رویدادهای رخداده را تشـــخیص داد. به‌عنوان مثال در نمای ۲، ۷۵۰ گره افزوده شده و ۷۳۸ یال افزوده شده است.

معیارهای مختلفی برای ارزیابی تشخیص جوامع به صورت پویا وجود دارد. یکـــی از ایـــن معیارها، معیار صحت اســـت. دو معیار پودمانگی و اطلاعات متقابل نرمال‌شـــده[9] برای ارزیابـــی صحت وجود دارد. اگر جوامع حقیقی موجود باشـــند از معیار اطلاعات متقابل نرمال‌شـــده اســـتفاده می‌شـــود. ولی اگر جوامع حقیقی موجود نباشند، از معیار پودمانگی می‌تـــوان بهره برد. بنابراین با توجه به موجود بودن جوامع حقیقـــی در مجموعهٔ داده‌ای دی‌بی‌ال‌پـــی، از معیار اطلاعات متقابل نرمال‌شده استفاده شده است.

معیـــار اطلاعات متقابل نرمال‌شـــده، میزان اطلاعات مشـــترک بین جوامع واقعی و جوامع تشخیص‌داده‌شـــده را مشخص می‌کند. هر چه این مشـــابهت بیشتر باشد، این معیار هم بیشتر می‌شود. در واقع این معیار نشـــان دهندهٔ صحت روش‌های تشخیص جوامع در شبکه‌های اجتماعی و در حالت پویا اســـت. معیار اطلاعات متقابل نرمال‌شـــده توسط وانگ [۱۵] به صورت فرمول (۱) به دست خواهد آمد:

$$NMI = \frac{-2\sum_{i,j} N_{i,j} log \frac{N_{i,j}N_t}{N_i N_j}}{\sum_i N_i log \frac{N_i}{N_t} + \sum_{i,j} N_j log \frac{N_j}{N_t}} \quad (۱)$$

9- NMI



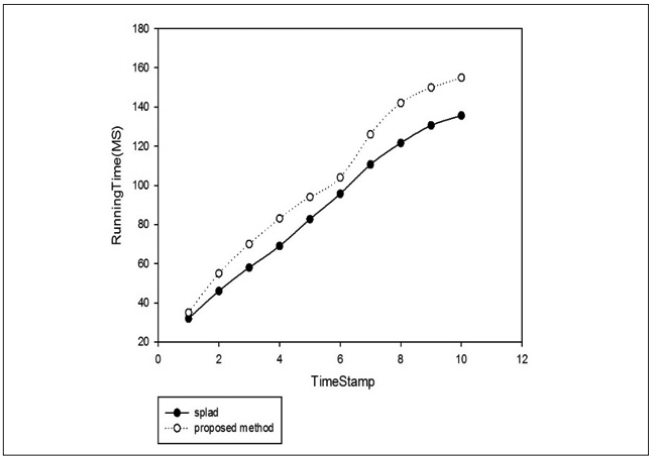
شکل ۱: اطلاعات متقابل نرمال‌شده روش ارائه‌شده و روش انتشار گوینده-شنونده پویا

N، ماتریس درهم‌ریختگی[10] است که بر روی جوامع تشخیص‌داده‌شده و جوامع واقعی تعریف می‌شـــود. هر کـــدام از درایه‌های این ماتریس یعنی $N_{(i,j)}$، تعداد گره‌های مشـــترک بین جامعهٔ تشخیص‌داده‌شدهٔ i و جامعهٔ واقعی j را مشـــخص می‌کند. N_i، مجموع مقادیر ســـطر iام ماتریس N و N_j، مجموع مقادیر سطر jام ماتریس N است. N_t، مجموع مقادیـــر کلیهٔ درایه‌های ماتریس N را نشـــان می‌دهد. معیار اطلاعات متقابل نرمال‌شده بین صفر تا یک است. بنابراین هر چه این مقدار به یک نزدیک‌تر باشـــد، جوامع تشخیص‌داده شده و جوامع واقعی به هم شبیه‌تر هستند.

در ادامه نتایج پیاده‌ســـازی روش ترکیبی ارائه‌شـــده و روش انتشـــار گوینده – شـــنونده بر روی مجموعهٔ داده‌ای دی‌بی‌ال‌پی در شکل (۱) آمده است. محور افقی شکل ۱، مشخص‌کننده‌ی شمارهٔ نما است که از ۱ تا ۱۰ اســـت. محور عمودی میزان معیار اطلاعات متقابل نرمال شده را نشان می‌دهد. این معیار هم بین ۰/۶۳ تا ۰/۷۷ است.

با توجه به شکل (۱) می‌توان گفت که روش ارائه شده از لحاظ میزان اطلاعات متقابل نرمال‌شده نسبت به روش انتشار گوینده-شنونده پویا بهتر عمل کرده اســـت. میزان اطلاعات متقابل نرمال‌شده برای هر دو

10- Confusion Matrix



شکل ۲: زمان اجرای روش ارائه‌شده و روش انتشار گوینده-شنونده پویا

روش در ابتدا تقریبا ۷۷ صدم است. در انتها یعنی در نمای آخر میزان این اطلاعات برای روش انتشـــار گوینده-شنونده پویا تقریبا ۶۳ صدم اســـت در حالی که برای روش ارائه‌شده، میزان این اطلاعات ۶۹ صدم است. بنابراین می‌توان گفت میزان کاهش اطلاعات متقابل نرمال‌شده در نماهای مختلف روش ارائه‌شـــده نسبت به انتشار گوینده-شنونده پویا کمتر است.

این موضوع نشـــان دهندهٔ این است که ترکیب روش انتشار برچسب گوینده - شـــنونده با روش بلاندل می‌تواند صحت تشـــخیص جوامع را در حالـــت پویا بهبود دهد. این بهبود در صحت تشـــخبص جوامع با توجه بـــه درنظرگرفتن اتصالات درونی جوامـــع و ارتباطات گره با جوامع جهت تشخیص این‌که آیا یک گره می‌تواند به جامعهٔ موردنظر بپیونـــدد یا خیر، رخ می‌دهد. در ادامه زمان اجـــرای دو الگوریتم در شـــکل (۲) نشان داده شده اســـت. محور افقی این شکل مانند شکل (۱) نشان‌دهندهٔ شماره نما از ۱ تا ۱۰ و محور عمودی این شکل نشان دهندهٔ زمان اجرا بر حســـب میلی‌ثانیه است. معیار زمان اجرا بین ۳۲ تا ۱۶۰ است.

همان‌طور که در شـــکل (۲) مشـــاهده می‌کنید، زمـــان اجرای روش ارائه‌شـــده و روش انتشـــار گوینده-شـــنونده پویا در نماهای مختلف تفاوت محسوســـی نســـبت به هم ندارند و تفاوت آن‌ها در حد چند میلی‌ثانیه اســـت. لذا با توجه به این‌که معیار سرعت که براساس زمان اجرا محاســـبه می‌شـــود، می‌توان گفت که ســـرعت دو روش تفاوت چندانی ندارد.

۵- نتیجه گیری

تشـــخیص جوامـــع در شـــبکه‌های اجتماعی در حالت پویـــا یکی از مســـائل مهم تحقیقاتی در سال‌های اخیر اســـت. در این مقاله، یکی از روش‌های مهم در این زمینه به نام روش انتشـــار گوینده - شنونده پویا بررسی شده و یکی از مشـــکلات آن شناسایی شده است. وجود این مشکل باعث کاهش صحت تشخیص جوامع شده بود. سپس یک روش ترکیبی بر مبنای روش انتشار گوینده-شنونده و روش حریصانهٔ بلاندل ارائه شده است. نتایج نشان می‌دهد که روش ارائه‌شده از لحاظ معیار صحت نســـبت به روش انتشار گوینده - شنونده پویا بهتر عمل کرده است. در عین حال سرعت روش ارائه‌شده با سرعت روش انتشار گوینده-شنونده پویا تفاوت خیلی محسوسی ندارد.

۷- مراجع

[1] H. Alvari, A. Hajibagheri, and G. Sukthankar, "Community detection in dynamic social networks: A game-theoretic approach," In Advances in Social Networks Analysis and Mining (ASONAM), IEEE/ACM International Conference, pp. 101–107, 2014.

[2] J. He and D. Chen, "A fast algorithm for com-community detection in temporal network," PhysicaA: Statistical Mechanics and its Applications, vol. 429, pp. 87–94, 2015.

[3] N. P. Nguyen, Y. Xuan, and M. T. Thai, «On detection of community structure in dynamic social ne-tworks,» Handbook of Optimization in Complex Networks, vol. 58, pp. 307-347, 2012.

[4] N. Aston, J. Hertzler, and W. Hu, "Overlapping community detection in dynamic networks," Journal of Software Engineering and Applications, vol. 7, pp. 872–882, 2014.

[5] J. Xie, M. Chen, and B. Szymanski, LabelRan-kT: "Incremental community detection in dyn-amic ne-tworks via label propagation," In Proceedings of the ACM Workshop on Dynamic Networks Management and Mining, pp. 25–32, 2013.

[6] G. W. Flake, S. Lawrence, and C. L. Giles, "Efficient identification of Web communities," In Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 150–160, 2000.

[7] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks.," In Proceedings of the national academy of sciences, vol. 99, pp. 7821–7826, 2002.

[8] J. Xie and B. K. Szymanski, "LabelRank: A stabilized label propagation algorithm for community detection in Networks," In Network Science Workshop (NSW), pp. 138-143, 2013.

[9] K. Liu, J. Huang, H. Sun, M. Wan, Y. Qi, and H. Li, "Label propagation based evolutionary clustering for detecting overlapping and non-overlapping communities in dynamic networks," Knowledge-Based Systems, vol. 89, pp. 487–496, 2015.

[10] S. H.L. and H. J.B. and T. Y.Q. and S.Q.B. and L.Huai-Liang, "Detecting overlapping communities in networks via dominant label propagation," Chinese Phys.B, vol. 24, p. 18703, 2015.

[11] J. Xie, B. K. Szymanski, and X. Liu, "SLPA: Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process," In Data Mining Workshops (ICDMW), pp. 344–349, 2011.

[12] V. D. Blondel, J. Guillaume, and E. Lefebvre, "Fast unfolding of communities in large networks," Journal of Statistical Mechanics: Theory and Experiment, vol. 10, pp. 1–12, 2008

[13] J. Thompson, «Community evolution in temporal networks» (Doctoral dissertation, Rensselaer Po-lytechnic Institute), 2015.

[14] Stanford large Dataset Network Collection, [Online], Availabe : http://www.snap.stanford.edu.

[15] L.Wang, «The structure and dynamics of large social networks» (Doctoral dissertation, Cornell University), 2013.

ترویج مدل بلوغ و پای‌بندی به استفاده از بهترین تجارب موجود زمینه ساز رشد تکاملی در ارائه خدمات فناوری اطلاعات

سید ابراهیم ابطحی
استادیار دانشکده مهندسی کامپیوتر- دانشگاه صنعتی شریف
پست الکترونیکی: abtahi@sharif.edu

طرح و واکاوی دشواری

مدتی است که برخی از خدمات اینترنتی لحظه‌ای که بر حسب مشخصات خود، تعدادی کاربر همزمان دارند، در آغاز برپایی خدمت با هجوم ناگزیر کاربران همزمان، با دشـــواری در ارائه خدمت مواجه شده و از ادامه ارائه خدمت باز می‌مانند که عموم مردم از این وضعیت با عبارت افتادن رایانه کارگزار، یاد می‌کنند. این اتفاق در این سال‌ها به دفعات تکرار شده، که دو نمونه اخیر آن، سرشماری رایانه‌ای اخیر مرکز آمار ایران و بلیط فروشی جشـــنواره فیلم فجر امسال بود. مشکل اول پس از مدتی با چاره‌اندیشی دست‌اندرکاران حل شد و سرشماری رایانه‌ای ادامه یافت اما دومی درست نشـــد و مردم علاقمند تهیه بلیط رایانه‌ای این جشـــنواره اکثرا موفق به تهیه بلیط اینترنتی نشـــدند و پس از ســـال‌ها انجام اینترنتی این اقدام، بدترین تجربه را در این مورد داشتند و حتی شرکت مجری در انجام روش دستی و به کمک رایانه‌ای و جبرانی این فعالیت نیز به علت کم‌تجربگی یا کم‌توانی، توفیقی نیافت. این تجربه برای علاقمندان به نقطه ضعف کلیدی جشنواره امسال تبدیل شد. در این نوشته به واکاوی این دشواری، علل و روش‌های جبرانی مدیریتی رایج می پردازیم.

راه‌حل‌های مدیریتی

دشـــواری مطروحه، به عنوان یک دشواری اجرائی، مشکل پیچیده ای نیست و راه‌حل‌های فنی متعددی از جمله استفاده از کارگزاران آینه‌ای دارد. اما این دشواری، بیشتر مشکلی مدیریتی است. زیرا در زمانی که این مشـــکل پیش آمد، همین گونه خدمـــات را مجریانی دیگر، تنها با تجاربی بیشـــتر، به خوبی ارائه کرده و می‌کنند. مســـئله اصلی انتخاب مجریان کم‌تجربه و کم‌توان و عدم اســـتفاده از بهترین تجارب موجود است که یک خطای مدیریتی در انتخاب مجری است. این اشتباه را با بهانه خدمتی در سبد حامی یا حامیان مالی جشنواره نمی‌توان توجیه کرد. درســـت اســـت که اصولا پروژه‌های فناوری اطلاعات و نرم‌افزار، پروژه‌های پرخطری هستند و بر اســـاس آمار ادواری مطالعه‌ای حدود نیمی از آن‌ها با شکســـت مواجه می‌شوند. اما این مورد، موردی رایج و نسبتا ساده با راه‌حل‌های معین و تجربه شده است. بنابراین با رعایت دو اصل رعایت ضوابط بلوغ در برون سپاری فعالیت‌ها و استفاده از بهترین تجارب موجود، در کارهای تجربه شـــده‌ای نظیر موضوع مورد بحث ما، به راحتی می‌شد از بروز این مشکلات و ایجاد نارضایتی در علاقمندان، جلوگیری کرد.

جمع‌بندی

بـــه بهانه ایـــن بحث، می‌توان به یک دشـــواری قابـــل حل در بخش انفورماتیک کشور اشـــاره کرد. این دشواری که به قلت تجربه و دانش برخی مدیران یا مشـــاوران آن‌ها بر می‌گردد و گاه از عدم اطلاع آن‌ها از وجود تجارب موجود موفق در این زمینه‌ها در کشور، ناشی می‌شود را با ترویج لزوم رعایت مدل بلوغ در برون‌ســـپاری و استفاده از بهترین تجارب موجود، می‌توان ســـامان داد و از تکرار مکرر اشتباهات پیشین، جلوگیری کرد و زمینه تکامل روز افزون ارائه خدمات لحظه‌ای اینترنتی را در کشور فراهم ساخت.

معرفی استاندارد ۱۵۹۴۰: خدمات محیط مهندسی نرم‌افزار

سیدعلی آذرکار
شرکت مهندسی پدیدپرداز
کمیسیون استاندارد و تدوین مقررات، سازمان نظام صنفی رایانه‌ای استان تهران
پست الکترونیکی: ali.azarkar@pdpsoft.com

۱- مقدمه

محیط‌های مهندسی نرم‌افزار (SoftwareEngineeringEnviroment-SEE) ناظر به مجموعه‌ای از خدمات است که به صورت کلی یا جزیی توسط ابزارهای نرم‌افزاری خودکار شده و برای پشتیبانی از اجرای فعالیت‌های انسانی در مهندسی سامانه‌ها و نرم‌افزار بکار گرفته می‌شود.

محیط مهندسی نرم‌افزار، می‌تواند در برگیرنده طیف گسترده‌ای باشد: از این‌که فقط دو یا چند ابزار روی یک سامانه عامل اجرا شدند تا محیط‌های کاملاً یکپارچه شده‌ای که مدیریت پایش و کنترل همه داده‌ها، فرآیندها و فعالیت‌های در چرخه‌حیات مهندسی سامانه‌ها و نرم‌افزار را ممکن می‌سازد.

به واسطه خودکارسازی فعالیت‌ها، چه کلی و چه جزیی، محیط مهندسی نرم‌افزار می‌تواند فواید زیادی را برای سازمان به همراه داشته باشد، که از آن جمله است کاهش هزینه‌ها، افزایش بهره‌وری و مدیریت بهتر (بهبود یافته) و در نهایت محصول با کیفیت‌تر.

پیچیدگی‌ها و تنوع خدماتی که سازمان‌های نرم‌افزاری برای توسعه محصولات نرم‌افزاری دارند از یک سو و نحوه تامین و استفاده از این خدمات از سوی دیگر، منجر به تدوین استاندارد ISO/IEC/IEEE 15940:2013 (با عنوان «مهندسی سامانه‌ها و نرم‌افزار- خدمات محیط مهندسی نرم‌افزار») شد، که در این گزارش به اختصار «استاندارد» نامیده می‌شود.

از آن‌جا که کلیه فرآیندهای مربوط به توسعه محصولات نرم‌افزاری در استاندارد ISO/IEC 2207:2008 (به اختصار: ۱۲۲۰۷) احصا و به تفصیل بیان شده، این استاندارد نیز به همین فرآیندها متکی است.

هدف این استاندارد، تعریف مجموعه‌ای از خدمات در محیط مهندسی نرم‌افزار است که با فرآیندهای استاندارد ۱۲۲۰۷ سازگاری داشته که می‌تواند هم به عنوان یک مرجع عمومی و هم به منظور تعریف یک فرآیند خودکار شده نرم‌افزاری (یا سامانه‌ای) استفاده شود.

سابقه این استاندارد به سال ۲۰۰۶ بر می‌گردد که نسخه اولیه آن تدوین شد. این استاندارد از سال ۲۰۱۰ تحت بازنگری قرار گرفت و نسخه جدید آن در سال ۲۰۱۳ منتشر شد.

مخاطبین بالقوه این استاندارد، می‌توانند افراد زیر باشند:

- کاربران و مهندسین سامانه نرم‌افزار
- تامین‌کنندگان ابزارها و محیط مهندسی نرم‌افزار
- کارفرمایان
- آموزش‌دهندگان مهندسی سامانه و نرم‌افزار
- مشاورین مهندسی سامانه و نرم‌افزار

جدول ۱: مشخصات استاندارد

شماره:	۱۵۹۴۰
قسمت:	-
نوع سند:	استاندارد بین‌المللی
عنوان لاتین:	Systems and software engineering - Software engineering environment services
عنوان فارسی:	مهندسی سامانه‌ها و نرم‌افزار – خدمات محیط مهندسی نرم‌افزار
تعداد صفحات:	۶۳
سال چاپ:	۲۰۱۳
زبان:	تک زبانه، انگلیسی
شماره ویرایش:	ندارد
متمم و اصلاحیه:	ندارد
استاندارد(های) جایگزین شده:	ISO/IEC 15940:2006
کد رده‌بندی استانداردی (ICS):	۳۵،۰۸۰
نهاد استانداردگذار:	ایزو
شماره استاندارد معادل فارسی:	-
ناشر استاندارد فارسی:	-
سال چاپ:	-
شماره ویرایش:	-
تعداد صفحات:	-

۲- مشخصات

مشخصات این استاندارد در جدول شماره ۱ نشان داده شده است.

۳- ساختار

ساختار این استاندارد به شرح زیر است:

- بند ۱: دامنه کاربرد
- بند ۲: اصطلاحات و تعاریف
- بند ۳: اصطلاحات مخفف
- بند ۴: مدل مرجع برای خدمات محیط مهندسی نرم‌افزار
- بند ۵: خدمات مهندسی نرم‌افزار
- بند ۶: خدمات مهندسی سامانه‌ها
- بند ۷: خدمات فنون مهندسی سامانه‌ها
- بند ۸: خدمات مدیریت فنی
- بند ۹: خدمات مدیریت پروژه
- بند ۱۰: خدمات مدیریت فرآیند
- بند ۱۱: خدمات پشتیبانی محیط مهندسی نرم‌افزار
- بند ۱۲: خدمات زیرساختی محیط مهندسی نرم‌افزار
- پیوست A: پشتیبانی خودکار نمونه برای خدمات محیط مهندسی نرم‌افزار
- پیوست B: نگاشت خدمات به فعالیت‌های ISO/IEC 12207
- پیوست C: نگاشت خدمات به فعالیت‌های ISO/IEC 15288
- پیوست D: ارتباط نمونه رده‌ها با خدمات محیط مهندسی نرم‌افزار
- پیوست E: کاربرد این استاندارد
- کتابشناسی

۴- مدل مرجع خدمات محیط مهندسی نرم‌افزار

این استاندارد یک مدل مرجع برای خدمات مهندسی نرم‌افزار ارائه می‌دهد. در این مدل از مجموعه‌ای از توصیف‌های مفهومی برای توصیف هر خدمتی که در محیط مهندسی نرم‌افزار به‌کار گرفته می‌شود، استفاده می‌کند. تاکید بر «توصیف مفهومی» از این منظر است که توصیف از یک «دیدگاه مرجع» بوده و با پیاده‌سازی‌های خاص سروکار ندارد. بر همین اساس، مدل ارائه شده مستقل از یک حوزه خاص کاربردی یا پروژه یا چرخه‌حیات یا ابزار بوده و بنابراین در هر سازمانی قابل استفاده است.

مجموعه خدماتی که در این استاندارد به تشریح آن پرداخته شده، به هشت رده تقسیم شده و در مجموع طیف گسترده‌ای از فعالیت‌های کارکردی را در یک سازمان نوعی مهندسی نرم‌افزار، منعکس می‌کند. این هشت رده عبارت است از:

۱- خدمات مهندسی نرم‌افزار (مانند مدل‌سازی نرم‌افزار)
۲- خدمات مهندسی سامانه‌ها (مانند و مدل‌سازی سامانه)
۳- خدمات فنون مهندسی سامانه‌ها (مانند تحلیل ارزش)
۴- خدمات مدیریت فنی (مانند استفاده مجدد، مدیریت پیکربندی)
۵- خدمات مدیریت پروژه (مانند برآورد، پایش پروژه)
۶- خدمات مدیریت فرآیند (مانند پایش فرآیند، بهبود فرآیند)
۷- خدمات پشتیبانی محیط مهندسی نرم‌افزار (مانند نشر، الزام خط‌مشی)
۸- خدمات زیرساختی محیط مهندسی نرم‌افزار (مانند انباره، ارتباطات، خدمات سامانه عامل)

۴-۱- ساختار توصیف خدمت

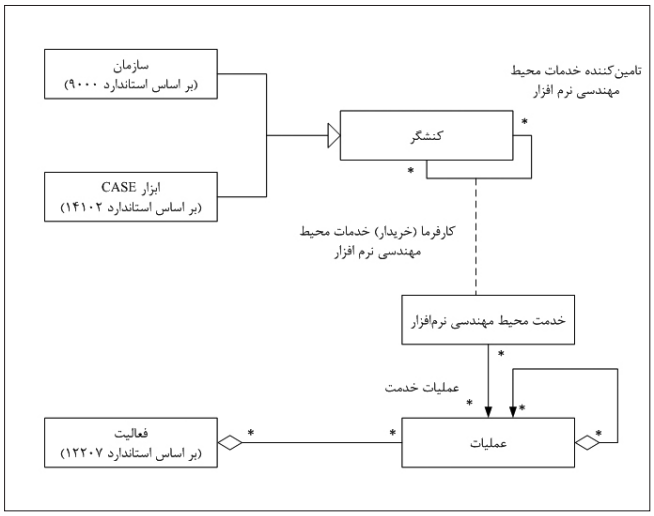
هر خدمت طبق این استاندارد ذیل دو سر فصل تعریف شده است:
- مفهـــوم خدمت: ارائه توصیفی از خدمت بـــا زبانی که مختص یک پیاده‌سازی خاص نباشد.
- عملیات خدمت: در برگیرنده عملیاتی است که ممکن است در خدمت گنجانده شده باشد. این فهرست از قابلیت‌های عملیاتی در اغلب موارد، فقط بیان‌کننده خدمات اصلی (یا اولیه) بوده و جامع نیست.

۴-۲- مدل مرجع

همان‌گونه که اشـــاره شد، خدمات محیط مهندسی نرم‌افزار می‌توانند در قالب یک «مدل مرجع» شناســـایی شود. این مدل در قالب UML در شکل شماره ۱ نشان داده شده است.
این مدل از مفاهیم زیر تشکیل شده است:
- محیط مهندسی سامانه و نرم‌افزار (خودِ مدل)
- خدمت محیط مهندســـی نرم‌افزار (شـــامل یک یـــا چند عملیات خدمت/ برای پشـــتیبانی فعالیت‌های چرخه حیات محیط مهندسی نرم‌افزار است.)
- عملیات خدمت در محیط مهندسی نرم‌افزار
- ابزار CASE: منظور اســـتفاده از رایانه در فرآیند مهندسی نرم‌افزار اســـت. ابزار CASE یک محصول نرم‌افزاری اســـت که به مهندســـین ســـامانه و نرم‌افزار از طریق ارائه پشتیبانی خودکار برای فعالیت‌های چرخه‌حیات مهندسی سامانه و نرم‌افزار، کمک می‌کند.
- کنشـــگر (Actor): ســـازمان یا ابزار CASE بوده که خدمات محیط مهندسی نرم‌افزار را تامین یا خریداری می‌کند.
- فعالیت
- سازمان: منظور گروهی از افراد و تسهیلات بوده که دارای چیدمانی از مسولیت‌ها، اختیارات و ارتباطات هستند.
در ادامه، خدمات پشـــتیبانی شده ذیل هر رده از رده‌های هشت‌گانه اشاره شده، آمده است.

۵- خدمات مهندسی نرم‌افزار

خدماتی که در ایـــن رده آمده، مربوط به خدمات خاص چرخه‌حیات نرم‌افزار است. خدمات تعریف شده در این رده عبارت است از:
- خدمات مهندسی نیازمندهای نرم‌افزار
- خدمات مهندسی معکوس نرم‌افزار
- خدمات بازمهندسی (مهندسی مجدد) نرم‌افزار
- خدمات نمونه‌سازی نرم‌افزار
- خدمات طراحی نرم‌افزار
- خدمات مدل‌سازی نرم‌افزار
- خدمات شبیه‌سازی نرم‌افزار
- خدمات تولید نرم‌افزار مبنی بر مولفه‌های نرم‌افزاری
- خدمات تولید متن کد نرم‌افزار



شکل ۱: مدل مرجع خدمات محیط مهندسی نرم‌افزار

- خدمات ترجمه نرم‌افزار
- خدمات خطازدایی نرم‌افزار
- خدمت تحلیل ایستا/پویای نرم‌افزار
- خدمت آزمون نرم‌افزار
- خدمت تصدیق نرم‌افزار
- خدمت یکپارچه‌سازی نرم‌افزار
- خدمت مهندسی حوزه نرم‌افزار (استفاده مجدد از نرم‌افزار)
- خدمت مدیریت اســـتفاده مجدد از دارایی نرم‌افزار (استفاده مجدد از نرم‌افزار)
- خدمت مدیریت اســـتفاده مجدد از برنامه نرم‌افزار (استفاده مجدد از نرم‌افزار)

۶- خدمات مهندسی سامانه‌ها

خدماتی که ذیل این رده آمده، مربوط به خدمات خاص چرخه‌حیات سامانه است. خدمات تعریف شده در این رده عبارت است از:
- خدمت جهت‌گیری (Orientation) راهکار سامانه
- خدمت سناریوهای عملیاتی سامانه
- خدمت مدل‌سازی سامانه
- خدمت طراحی معمارانه (کلان) سامانه
- خدمت مهندسی نیازمندی‌های سامانه
- خدمت مهندسی مجدد سامانه
- خدمت شبیه‌سازی سامانه
- خدمت یکپارچه‌سازی سامانه
- خدمت آزمون سامانه
- خدمت تحلیل و گزارش‌دهی آزمون سامانه
- خدمت تصدیق محصولات کاری سامانه

۷- خدمات فنون مهندسی سامانه

خدماتی که ذیل این رده آمده، شامل مهندسی و مدیریت پروژه است. این خدمات به فعالیت‌هایی مرتبط اســت که معمولاً توسط مدیران و مهندسین به اشــتراک گذاشته می‌شود. خدمات تعریف شده در این رده، عبارت است از:

- خدمت تحلیل ارزش
- خدمت تحلیل توازن (Trade off)
- خدمت تحلیل اثربخشی
- خدمت تحلیل بلوغ فناوری

۸- خدمات مدیریت فنی

خدماتی که ذیل این رده قرار گرفته، هم مربوط به مهندسی سامانه‌ها و نرم‌افزار و هم مدیریت پروژه اســت. ایــن مجموعه خدمات مربوط به فعالیت‌هایی اســت که اغلب بین مهندسین و مدیران به اشتراک گذاشته می‌شود. خدمات تعریف شده در این رده عبارت است از:

- خدمت مدیریت پیکربندی
- خدمت مدیریت تغییر
- خدمت مدیریت انباره محیط مهندسی نرم‌افزار
- خدمت مدیریت استفاده مجدد
- خدمت سنجش و تحلیل
- خدمت تضمین کیفیت
- خدمت ممیزی
- خدمت ردیابی نرم‌افزار
- خدمت مستندسازی
- خدمت پشتیبانی بازنگری

۹- خدمات مدیریت پروژه

خدمات ذیل ایــن رده، برای پشــتیبانی از فعالیت‌های طرح‌ریزی و اجرای یک پروژه اســت. پس از شــروع پــروژه، طرح‌ریزی تفصیلی فعالیت‌هــا در کنار پایش و طرح‌ریزی مجدد فعالیت‌های پروژه، برای حصول اطمینان از پیشــرفت مستمر آن، الزامی خواهد بود. خدمات تعریف شده در این رده عبارت است از:

- خدمت راهبرد پروژه
- خدمت طرح‌ریزی پروژه
- خدمت برآورد پروژه
- خدمت مدیریت مخاطره پروژه
- خدمت پایش و زمان‌بندی پروژه
- خدمت ارزشیابی پروژه
- خدمت مدیریت تصمیم‌گیری پروژه
- خدمت مدیریت اطلاعات

۱۰- خدمات مدیریت فرآیند

خدمات ذیل این رده، برای پشتیبانی از پروژه‌ها در دستیابی به نظم، کنترل و درک روشــن آن‌ها از فرآیندهای توسعه نرم‌افزار و گام‌های فرآیندی (آن‌گونه که در اســتاندارد ۱۲۲۰۷ آمده) بیان شــده است. خدمات تعریف شده در این رده عبارت است از:

خدمت تعریف فرآیند
خدمت کتابخانه فرآیند
خدمت شروع فرآیند
خدمت به‌کارگیری فرآیند
خدمت پایش فرآیند
خدمت پشتیبانی بهبود فرآیند
خدمت مستندسازی فرآیند

۱۱- خدمات پشتیبانی محیط مهندسی نرم‌افزار

خدماتی که ذیل این داده به آن‌ها پرداخته شده، آن‌هایی هستند که دیگر خدمات برای عملیاتی شدن نیازمند آن‌ها هستند. این خدمات به شــکل عمومی مرتبط با پردازش، قالب‌بنــدی، و امحای داده‌ها و اطلاعات قابل خواندن توسط انسان است. خدمات تعریف شده در این رده، عبارت است از:

- خدمت عمومی پشتیبانی محیط مهندسی نرم‌افزار
- خدمت انتشار (نشر) محیط مهندسی نرم‌افزار
- خدمت پشتیبانی کار مشارکتی در محیط مهندسی نرم‌افزار
- خدمت پشــتیبانی از اطلاع‌رســانی کاربری در محیط مهندســی نرم‌افزار
- خدمت راهبری محیط مهندسی نرم‌افزار
- خدمت الزام خط‌مشی محیط مهندسی نرم‌افزار
- خدمت کاوش (Mining) داده‌ها/اطلاعات محیط مهندسی نرم‌افزار
- خدمت ذخیره‌سازی/بازاریابی داده‌های محیط مهندسی نرم‌افزار
- خدمت مبادله داده‌ها/اطلاعات محیط مهندسی نرم‌افزار
- خدمت پشتیبان توان‌ساز محیط مهندسی نرم‌افزار

۱۲- خدمات زیرساختی محیط مهندسی نرم‌افزار

خدماتی که ذیل این رده به آن‌ها اشــاره شده، مربوط به فعالیت‌های زیرساختی یک محیط مهندســی نرم‌افزار بوده که برای پشتیبانی از پیاده‌ســازی واقعی (یا اجرایی) محیط مهندســی نرم‌افزار، مورد نیاز خواهد بود. خدمات تعریف شده در این رده، عبارت است از:

- خدمت مدیریت زیرساخت محیط مهندسی نرم‌افزار
- خدمت اشتراک‌گذاری اطلاعات محیط مهندسی نرم‌افزار
- خدمت انباره محیط مهندسی نرم‌افزار
- خدمت سامانه عامل محیط مهندسی نرم‌افزار

۱۳- مثال‌هایی از تعریف خدمات

در متن اصلی استاندارد، برای تمامی خدماتی که ذیل هر رده برشمرده شد، دو سرفصل «مفهوم خدمت» و «عملیات خدمت» تعریف شده است. به منظور پرهیز از طولانی شدن گزارش، تمامی این سرفصل‌ها توضیح داده نشده است. با این حال، برای آشنایی خواننده با ساختار کلی استاندارد، در این بخش سعی شده از هر رده یک خدمت انتخاب شده و دو سر فصل فوق برای آن‌ها بیان شود.

۱۳-۱- «خدمت طراحی نرم‌افزار»

این خدمت ذیل رده «خدمات مهندسی نرم‌افزار» تعریف شده است.

۱۳-۱-۱- مفهوم خدمت

این خدمت توانایی ثبت، نمایش، تحلیل، صحه‌گذاری و پالایش آن دسته از نیازمندی‌های سامانه که نیازهای عملیاتی را برآورده ساخته و بر عهده مولفه‌های نرم‌افزاری گذاشته شده، فراهم می‌کند.

۱۳-۱-۲- عملیات خدمت

این خدمت قابلیت‌های زیر را فراهم می‌کند:

- کشف (شناسایی) و ثبت نیازمندی‌های نرم‌افزاری و کسب‌وکار
- ساختاردهی نیازمندی‌های نرم‌افزاری
- ایجاد، اصلاح، مرور و نمایش نیازمندی‌های نرم‌افزار
- گروه‌بندی و اولویت‌بندی نیازمندی‌های نرم‌افزاری
- بررسی سازگاری نیازمندی‌های نرم‌افزاری
- اختصاص نیازمندی‌های نرم‌افزار به هر مولفه نرم‌افزاری
- اجرای تحلیل تاثیرات ناشی از اضافه، کاهش، یا اصلاح یک نیازمندی بر پروژه، ارزش، منابع و زمان‌بندی
- صحه‌گذاری و مبناگذاری مستند مشخصه بر اساس (نظرات) ذی‌نفعان و توسعه‌دهندگان

۱۳-۲-«خدمت طراحی کلان (معمارانه) سامانه»

این خدمت ذیل رده «خدمات مهندسی سامانه‌ها» تعریف شده است.

۱۳-۲-۱- مفهوم خدمت

این خدمت توانایی تعریف دیدگاه‌های حیاتی، تعریف پیش‌ران‌های معماری و یافتن بهترین معماری را برای پشتیبانی تحلیل نیازهای عملیاتی و مدل‌های معماری، ایجاد می‌کند.

۱۳-۲-۲- عملیات خدمت

این خدمت قابلیت‌های زیر را فراهم می‌کند:

- تعریف دیدگاه‌های معمارانه (یعنی کارایی اختصاص مولفه‌های کارکردها/ معمارانه، پیچیدگی واسط، زنجیره کارکردهای حیاتی)
- شناسائی پیش‌ران‌های معماری (یعنی انعطاف‌پذیری در قبال تکامل عملیاتی یا فنی، هزینه، استفاده مجدد از دارایی‌های قدیمی، لجستیک و غیره)
- تحلیل مدل‌ها طبق دیدگاه‌ها (به عنوان مثال، بر اساس دیدگاه‌های امنیتی)
- اعمال قواعد معماری (به عنوان مثال، مبتنی بر موتور قواعد، الگوهای معماری)

۱۳-۳- تحلیل اثربخشی

این خدمت ذیل رده «خدمات فنون مهندسی سامانه» آمده است.

۱۳-۳-۱- مفهوم خدمت

این خدمت توانایی ارائه یک ارزیابی از این‌که چگونه از یک محصول مرتبط با یک راهکار طراحی منطقی یا فیزیکی جایگزین، انتظار می‌رود با فرض یک سناریوی پیش‌بینی شده برای به‌کارگیری، اجرا شده و عمل کند.

۱۳-۳-۲- عملیات خدمت

این خدمت توانایی‌های زیر را فراهم می‌کند:

- ارزیابی محصول در مقایسه با سناریوی به‌کارگیری
- پایش اثربخشی
- ثبت سناریوی به‌کارگیری و اثربخشی محصول

۱۳-۴- «خدمت تضمین کیفیت»

این خدمت ذیل رده «خدمات مدیریت فنی» آمده است.

۱۳-۴-۱- مفهوم خدمت

طبق مجموعه‌ای از اهداف پروژه که مرتبط با محدودیت‌های یک پروژه است، را مجاز می‌سازد.

۱۳-۴-۲- عملیات خدمت

این خدمت قابلیت‌های زیر را فراهم می‌کند:

- ترجمه (تبدیل) اهداف پروژه به رویدادهای مهم پروژه
- کمی‌کردن ورودی‌ها و خروجی‌ها برای فعالیت‌های کاری
- امکان انتصاب ارتباطات (وابستگی‌ها) بین فعالیت‌ها
- پیش‌بینی زمان‌های پیش‌افتادگی رویداد
- محاسبه تاریخ‌های شروع و پایان
- تحلیل مسیرهای بحرانی
- تولید زمان‌بندی‌های تفصیلی رویداد
-امکان تخصیص منابع
- ثبت واگذاری مسئولیت‌های کاری به افراد یا سازمان‌ها
- ایجاد کانال‌های اطلاع‌رسانی درون تیم پروژه (شامل شناسایی رسانه اطلاع‌رسانی، بسامد و محتوا)
- ایجاد معیارهای ارزشیابی

۱۳-۵- «خدمت تعریف فرآیند»

این خدمت ذیل رده «خدمات تعریف فرآیند» تعریف شده است.

۱۳-۵-۱- مفهوم خدمت

این خدمت ایجاد فرآیندهای سازمانی که چرخه‌حیات مهندسی سامانه و نرم‌افزار را پوشش می‌دهد، پشتیبانی می‌کند. این کار از طریق اقتباس و مناسب‌سازی مجموعه‌ای از فرآیندهای سطح بالاتر

جدول ۲: ارتباط با فرآیندهای استاندارد ۱۲۲۰۷

فرآیند در استاندارد ۱۲۲۰۷			خدمات محیط مهندسی نرم‌افزار
پیاده‌سازی فنی	فرآیند تعریف نیازمندی‌های سودبر	شناسایی سودبر	خدمت طرح‌ریزی پروژه
		شناسایی نیازمندی‌ها	خدمات مهندسی نیازمندی‌ها
		ارزشیابی نیازمندی‌ها	خدمات مهندسی نیازمندی‌ها
		توافق نیازمندی‌ها	خدمات مهندسی نیازمندی‌ها
		ثبت نیازمندی‌ها	خدمات مهندسی نیازمندی‌ها

انجام می‌شود. این اقتباس شامل ویژگی‌های خط‌مشی‌های سازمانی، فناوری زیرساختی، روش‌ها و رویه‌ها است.

۱۳-۵-۲- عملیات خدمت

این خدمت قابلیت‌های زیر را فراهم می‌کند:

- تحلیل نیازمندی‌های فرآیند، شامل تحلیل خاص حوزه و تحلیل خاص برنامه کاربردی
- نمونه‌سازی، ترکیب، تفکیک، مناسب‌سازی یا ماجولار کردن تعاریف فرآیند
- شبیه‌سازی، مدل‌سازی و صحه‌گذاری تعاریف فرآیند در مقایسه با راهنماها، استانداردها و خط مشی‌های سازمان

۱۳-۶- «خدمت ذخیره/بازیابی داده‌ها»

این خدمت ذیل رده «خدمات پشتیبان محیط مهندسی نرم‌افزار» تعریف شده است.

۱۳-۶-۱- مفهوم خدمت

این خدمت توانایی اداره کردن ذخیره/بازیابی داده‌ها را فراهم می‌کند.

۱۳-۶-۲- عملیات خدمت

این خدمت توانایی‌های زیر را فراهم می‌کند:

- بازیابی داده‌های اصلی
- بررسی حقوق داده‌ها/اطلاعات
- ثبت سوابق داده‌ها
- نگهداری داده‌ها و سوابق ذخیره طبق بازه‌های مشخص شده ذخیره و بازیابی و نیز خط‌مشی، توافق‌نامه‌ها و قوانین سازمان

۱۳-۷- خدمت مدیریت زیرساخت محیط مهندسی نرم‌افزار

این خدمت ذیل رده «خدمات زیرساخت محیط مهندسی نرم‌افزار» آمده است.

۱۳-۷-۱- مفهوم خدمت

این خدمت توانایی منابع زیرساختی محیط مهندسی نرم‌افزار را فراهم می‌کند.

۱۳-۷-۲- عملیات خدمت

- اضافه، حذف یا اصلاح منابع (از جمله ابزارها)
- پرس‌وجوی (Query) وضعیت و ارایه یک منبع
- ایجاد امکان دسترسی یک کاربر خاص و یا رده‌ای از کاربران (که بر حسب نقش تعریف شده) به یک منبع
- ارسال پیام‌های تشخیصی (Diagonistics) به راهبر سامانه
- تبادل داده‌ها یا ابزارها بین محیط‌های مهندسی نرم‌افزار
- انتقال یک کاربر و/یا رده‌هایی از کاربران (که بر حسب نقش تعریف شده) بین محیط‌های مهندسی نرم‌افزار
- انتقال توصیف‌های کار بین محیط‌های مهندسی نرم‌افزار
- برپاسازی، جمع‌آوری، ثبت‌نام یا حذف ثبت‌نام ابزارها و به روزرسانی آن‌ها به یک نسخه جدید
- پایش و تشخیص تسهیلات محیط مهندسی نرم‌افزار

۱۴- مثال‌هایی از پشتیبانی‌های خودکارشده برای هر خدمت

پیوست A این استاندارد، مثال‌هایی از پشتیبانی‌های خودکار برای هر خدمت در محیط مهندسی نرم‌افزار را ارائه کرده که به خواننده در درک موضوع و نحوه بکارگیری انواع ابزارهای احتمالی کمک می‌کند. به عنوان مثال، پشتیبانی‌های خودکار شده برای خدمت «مهندسی نیازمندی‌های نرم‌افزار»، می‌تواند به شرح زیر باشد:

- پشتیبانی قابلیت ردیابی نیازمندی‌ها
- پشتیبانی مدیریت انباره (نیازمندی‌ها)
- پشتیبانی اصلاح و پرس‌وجوی نیازمندی‌ها
- پشتیبانی ارائه نیازمندی‌ها
- پشتیبانی بررسی سازگاری
- پشتیبانی تحلیل تاثیر
- پشتیبانی صحه‌گذاری نیازمندی‌ها

۱۵- ارتباط با فرآیندهای ISO/IEC 12207 (نسخه ۲۰۰۸)

همان‌گونه که پیشتر اشاره شد، خدمات پیش‌بینی شده در این استاندارد، مبتنی بر فرآیندهایی بوده که برای فرآیندهای چرخه‌حیات نرم‌افزار بیان و تشریح شده است. در پیوست B این استاندارد، این ارتباط توضیح داده شده است. به عنوان مثال، جدول شماره ۱، ارتباط بین یک فرآیند نمونه از استاندارد ۱۲۲۰۷ را با خدمات پیش‌بینی شده در این استاندارد نشان می‌دهد.

جدول ۳: ارتباط با فرآیندهای استاندارد ۱۵۲۸۸

درخواست محیط مهندسی نرم‌افزار	فرآیندهای نظیر در استاندارد ۱۵۲۸۸		
خدمات طرح‌ریزی پروژه	توان‌سازی پروژه سازمانی	فرآیند مدیریت منابع انسانی	شناسایی مهارت‌ها
خدمات طرح‌ریزی پروژه			توسعه مهارت‌ها
خدمت شروع پروژه			توسعه مهارت‌ها
خدمت پشتیبانی از اطلاع‌رسانی کاربر محیط مهندسی نرم‌افزار			خرید و تامین مهارت‌ها
خدمت پشتیبانی از کار مشارکتی در محیط مهندسی نرم‌افزار			اجرای مدیریت دانش
خدمت تضمین کیفیت		فرآیند مدیریت کیفیت	طرح‌ریزی مدیریت کیفیت
خدمت پایش و زمان‌بندی پروژه			ارزیابی مدیریت کیفیت
خدمت پایش فرآیند			اجرای اقدام اصلاحی مدیریت کیفیت

۱۶- ارتباط با فرآیندهای ISO/IEC 15288 (نسخه ۲۰۰۸)

در پیوست C استاندارد، ارتباط آن با استاندارد ۱۵۲۸۸ نشان داده شده است. جدول شماره ۲ که برگرفته از این پیوست است، نگاشتی از فرآیندهای این استاندارد را به فرآیندهای این استاندارد نشان می‌دهد.

۱۷- قدردانی

از سرکار خانم مرجان طاهری که زحمت تهیه و آماده‌سازی پیش‌نویس این مقاله را متقبل شدند، سپاسگزاری می‌کنم.

درس‌هایی از کارِ برنامه‌نویسان

علیرضا خلیلیان
دانشجوی دکتری نرم‌افزار، گروه مهندسی نرم‌افزار، دانشکده مهندسی کامپیوتر، دانشگاه اصفهان
پست الکترونیکی: khalilian@eng.ui.ac.ir

چکیده

اگر تاکنون فرصت نکرده‌اید کتاب «کارِ برنامه‌نویسان» را بخوانید، اشکالی ندارد؛ به شما توصیه می‌کنم چند دقیقه‌ای را با ما در این مقاله بگذرانید تا جان کلامِ این کتاب ۵۳۶ صفحه‌ای را تقدیم شما کنیم. کسی چه می‌داند؛ شاید شما هم راغب شدید کل کتاب را بخوانید. دونالد کنوث در شاهکار چند جلدی‌اش، مقاله‌های قوی را خوانده است، آن‌ها را در ذهن خودش هضم کرده است و حاصل کار را در چندین خط یا پاراگراف خلاصه بیان نموده است. ما هم همین‌کار را با کتاب «کار برنامه‌نویسان» کردیم! چه مقایسه‌ای! با استاد ممتاز دانشگاه استنفورد و نابغه الگوریتم. بگذریم. با ما در این مقاله همراه باشید تا نکات برجسته را برای شما گزارش کنیم.

مقدمه

کتاب «کار برنامه‌نویسان» [۱] در سال ۲۰۰۹ توسط پیتر سیبِل نوشته شده است. این کتاب توسط آقای ابراهیم نقیب‌زاده مشایخ ترجمه شده و متن آن به‌تدریج در شماره‌هایی از مجله «گزارش کامپیوتر» به‌چاپ رسیده است. همچنین در سال ۱۳۹۱ به سفارش انجمن انفورماتیک ایران [1] چاپ شده است. کتاب ضمن این‌که ترجمه بسیار روانی دارد، خود شمول است؛ به این معنی که مترجم هر جا اصطلاح خاصی ذکر شده توضیح اضافه‌ای برای درک آن درون متن درج کرده است که خواننده را از جست‌وجو به دنبال معنی و مفهوم آن بی‌نیاز سازد.

این کتاب گزارش مصاحبه‌هایی است که پیتر سیبل با ۱۵ نفر از برنامه‌نویسان مشهور انجام داده است. نویسنده کتاب سعی کرده مصاحبه‌هایش را به‌طور نظام‌یافته ترتیب دهد و تقریباً سؤال‌های مشابهی از همه پرسیده است. این اقدام خوب، زمینه‌ای را فراهم کرده که بتوانیم با خواندن کتاب و برجسته کردن نکات مهم، درس‌هایی از سال‌ها فعالیت و تجربه برنامه‌نویسان بیاموزیم. بسیاری از درس‌ها و نکات ممکن است در کتاب‌های آکادمیک یافت شوند ولی وقتی در خلال شرح حالِ کار و زندگی فردی مشهور بیان می‌شوند، بهتر به دل می‌نشینند و درک می‌شوند. این نکات بخصوص مهم هستند چون افراد مصاحبه شونده اغلب از زمانی که فقط زبان‌های اولیه‌ای همچون اسمبلی و فرترن و رایانه‌های اولیه‌ای که تنها چند صد بایت حافظه داشته‌اند شروع به‌کار کرده‌اند و امروز که رایانه‌های چند هسته‌ای با ده‌ها ترابایت حافظه و صدها زبان[۲] برنامه‌سازی داریم برای ما حرف می‌زنند.

به‌همین دلیل نویسنده مقاله حاضر موقع خواندن هر یک از ۱۵ فصل، نکات مهم را یادداشت‌برداری کرد تا بعد از خلاصه‌سازی و پالایش نکات، آن‌ها را در قالب مقاله حاضر ارائه نماید. هنگام نکته‌برداری سعی شده نکاتی برجسته شوند که حتی‌الامکان همه مصاحبه‌شوندگان به‌نوعی به آن پاسخ داده‌اند که امکان مقایسه فراهم گردد. همچنین نکاتی برداشت شده‌اند که در حال حاضر معتبر و قابل استفاده باشند یا درس مهمی برای فعالیت‌های امروز و آینده برنامه‌نویسان داشته باشند. علاوه بر نقل نکته‌ها، هر جا که لازم بوده،

1- http://isi.org.ir/index.asp

۲- به وبگاه http://www.99-bottles-of-beer.net/ مراجعه کنید تا کدی با عملکرد یکسان را به ۱۵۰۰ زبان برنامه‌سازی مختلف ببینید.

مطالب بیشـــتری هم برگرفته از مقاله‌ها یا کتاب‌های معتبر امروزی اضافه شده‌اند که سودمندی مطالب نقل شده را بیشتر نمایند.

مصاحبه‌شوندگان

اغلب مصاحبه‌شـــوندگان در دانشـــگاه‌های معتبری چـــون هاروارد، استنفورد و ام‌آی‌تی در رشـــته‌هایی مثل ریاضی، فیزیک و مهندسی برق در ســـطوح مختلف از کارشناسی تا دکتری تحصیل کرده‌اند. این افراد عموماً در شـــرکت‌های معتبری مثل گوگل، مایکروسافت و یاهو کار کرده‌اند یا مشغول به‌کارند. همچنین اغلب آن‌ها از اعضای دائمی یا برجسته IEEE، ACM یا فرهنگستان ملی مهندسی هستند. افرادی که پیتر سیبل برای مصاحبه انتخاب کرده است عبارتند از:

جیمی زاوینسکی: نویسنده نت‌اِسکیپ
براد فیتز پاتریک: پدید آورنده لایو ژورنال
داگلاس کراکفورد: معمار ارشد جاوا اِسکریپت
برندن اِیک: خالق جاوا اِسکریپت
جاشوا بلاک: معمار ارشد جاوا
جو آرمسترانگ: خالق زبان اِرلانگ
سایمون پیتون جونز: معمار زبان هاسکِل
پیتر نورویگ: مدیر بخش پژوهش شرکت گوگل
گای استیل: طراح زبان اِسکیم
دن اینگالس: پیاده‌ساز اِسمال‌تاک
پیتر دویچ: نویسنده گوست اِسکریپت
کن تامپسون: خالق سیستم عامل یونیکس
فران آلن: پیشگام مترجم‌های بهینه‌ساز
برنی کاسل: استاد اشکال‌زدایی
دونالد کنوث: نویسنده کتاب هنر برنامه‌نویسی رایانه

از بین این ۱۵ نفر تنها یک‌نفر یعنی فران آلن خانم است که نخستین خانمـــی بود که عنوان «همکار آی‌بی‌ام »[3] کـــه بالاترین عنوان فنی افتخاری در آی‌بی‌ام اســـت را به‌دســـت آورد. اغلب مصاحبه‌شوندگان یک یا چند جایزه از جوایز معتبر زیر را به پاس نوآوری‌ها یا ســـال‌ها فعالیت و تجربه دریافت کرده‌اند:

جایزه تورینگ ACM: سالانه به اشخاصی که سهم به‌سزایی در زمینه رایانه دارند، اعطا می‌شود. شرکت‌های گوگل و اینتل حامیان مالی این جایزه هستند و معادل نوبل در نظر گرفته می‌شود.

جایزه جولت[4]: جایزه‌ایســـت که هر ســـال به محصولاتی داده می‌شود کـــه اهمیت فوق‌العاده آن‌ها باعـــث تکانی در صنعت و منجر به تولید سریعتر، ساده‌تر و کارامدتر نرم‌افزار شوند.

جایزه دستاوردهای استثنائی ناسا[5]: جایزه‌ایست که به افراد غیرنظامی ناســـا و فضانوردان اعطا می‌شود. فرد دریافت کننده جایزه، کارمندی از ناساســـت که به نوآوری مهمی دســـت‌یافته اســـت که باعث بهبود چشـــمگیر عملیات، کارایی، خدمات، صرفه‌جویـــی اقتصادی، علم یا فناوری شده است که مستقیماً سهمی در مأموریت ناسا داشته باشد.

جایزه گریس هاپر ACM[6]: جایزه‌ایست که به متخصص رایانه‌ای داده می‌شود که نوآوری مهم فنی یا خدماتی تا ۳۵ سالگی داشته باشد.

جایزه تعالی در برنامه‌نویســـی دکتر داب[7]: جایزه‌ایســـت که هر ساله به افرادی که از نظر ویراســـتاران مجلّهٔ دکتر داب، ســـهم بسزایی در پیشرفت توسعه نرم‌افزار داشته باشد اعطا می‌گردد.

جایزه سیستم نرم‌افزاری ACM[8] : این جایزه سالانه به افراد یا سازمانی اعطا می‌شود که سیســـتم نرم‌افزاری توسعه داده‌اند که تأثیر طولانی مدتی داشته است که این تأثیر در پذیرش تجاری یا کمک به مفاهیم یا هر دو منعکس شده باشد.

جایزه تیســـوتومو کانائی IEEE[9]: این جایزه به فردی که نوآوری‌های اساســـی در سیستم‌های رایانش توزیعی و کاربردهای آن داشته باشد اعطا می‌گردد.

هوش بالا، تجربه طولانی، شـــهرت و دریافت جوایز معتبر بین‌المللی باعث می‌شـــوند بتوانیم حرف‌های مصاحبه‌شوندگان را تعمیم دهیم و آن‌ها را بـــرای زمان‌ها، مکان‌ها و موقعیت‌های گوناگون معتبر بدانیم. اغلب مصاحبه‌شوندگان با کامپیوتر در مدرسه و دبیرستان در دهه‌های ۵۰ تا ۷۰ میلادی آشـــنا شده‌اند و اکثر آن‌ها بخصوص قدیمی‌ترها با انواعـــی از کامپیوترهای آی‌بی‌ام و زبان فرترن ســـال‌ها کار کرده‌اند. یکـــی از عوامل موفقیت زبان‌ها در کتاب‌های طراحی و پیاده‌ســـازی زبان‌ها [۲, ۳] حمایت مالی و نفوذ شـــرکت سازنده[10] ذکر شده است. اکنون بهتر روشـــن می‌شـــود که چرا زبانی مثل فرترن آن‌قدر مورد اســـتفاده همگان بوده در حالی‌که زبانی مثل الگول که طراحی بسیار زیباتری داشته است دوام چندانی نداشته است. فرترن از دل آی‌بی‌ام غول محاســـبات بیرون آمد و همه سازمان‌ها و دانشگاه‌ها کم و بیش رایانه‌هـــای آی‌بی‌ام را خریداری می‌کردند که روی آن‌ها فرترن نصب بوده است. درحالی‌که الگول زبانی است دانشگاهی و موقعیتی مشابه فرترن نداشـــته است. جو آرمســـترانگ هم در مورد زبان ارلانگ بیان کرده که شاید بهتر باشد مایکروسافت برخی از ایده‌هایش را بگیرد و در قالب محصولی عرضه کند که ارلانگ هم بازار وسیعی پیدا کند.

خوانایی کد

تقریبـــاً همه افراد بـــر اهمیت خوانایی کد تأکیـــد دارند. خوانایی کد تأثیر اساسی بر سهولت اشکال‌زدایی، کم کردن هزینه‌ها و فعالیت‌های مرحله نگهداری کد و افزایش قابلیت اطمینان نرم‌افزار دارد. استفاده از شناســـه‌های طولانـــی برای خوانایی توســـط اکثر افـــراد همچون

3- IBM Fellow
4- Jolt
5- NASA Exceptional Achievement Medal
6- ACM's Grace Murry Hopper
7- Dr, Dobb's Excellence in Programming
8- ACM Software System
9- IEEE's Tsutomu Kanai
10- Patronage, Inertia, and Sponsorship

زاوینسکی و بلاک توصیه شده است. آرمسترانگ سعی کرده از اسامی بسیار نامتشابه اســتفاده کند تا ابهام از بین رود. گذاشتن توضیحات در متن برنامه توســط زاوینسکی لازم دانسته شده ولی بعضی توصیه نمی‌کنند. آرمسترانگ مستندسازی کد و نوشتن مشخصات کد را در حد معقول لازم می‌داند. بِلاک خوانا بودن برنامه و هنر نویســندگی و کار ادبیاتی در کد نویسی را خیلی مهم می‌داند. اکثریت بر این باورند که کد خوانا مربوط به برنامه‌نویسی است که نویسنده خوبی هم باشد.

زبان‌های برنامه‌سازی

اکثر مصاحبه‌شــوندگان با زبان‌های معدودی کار کرده‌اند ولی مواردی هم هســت که زبان‌های بسیاری مورد استفاده قرار گرفته‌اند. اسمبلی، فرترن و لیســپ در زمره زبان‌هایی هســتند که مورد استفاده اکثریت بوده‌اند. جاوا، پایتون، پرل و جاوا اســکریپت هم ســایر زبان‌های مورد اســتفاده بوده‌اند. تعدادی هم از C و از C++ استفاده کرده‌اند با علم به این‌که این دو زبان مشکلات امنیتی بسیاری دارند. کاسل اظهار داشته که باوجود مشکلات امنیتی بسیار، C نعمت بزرگی برای برنامه‌نویسی سیســتم بوده است. اغلب افراد مثل آرمســترانگ رابطه خوبی با C++ ندارند زیرا آن‌را زبانــی فوق‌العاده بزرگ و پیچیده می‌دانند. باید توجه کرد که C و C++ زبان‌هایی هستند که اساساً برای کارایی و برنامه‌سازی سیســتم طراحی شده‌اند. دو ویژگی مذکور طراح را مجبور می‌کند که هــر نوع محدودیتی را بردارد و قابلیت‌های متعدد را در زبان قرار دهد. در این‌صورت همه چیز به‌عهده برنامه‌ســاز اســت و زبان به هیچ وجه مناسب افراد غیر حرفه‌ای نیســت. به گفته بلاک، برنامه‌سازانی که از C++ استفاده می‌کنند، معمولاً زیرمجموعه‌ای از آن را انتخاب می‌کنند. این موضوع و پیچیدگی زیاد، باعث می‌شود که تامپسون اظهار دارد از C++ خوشــش نمی‌آید و باوجود این‌که C++ یکی از چهار زبان گوگل است ولی اصلاً برای انتقال الگوریتم مناسب نیست.

افرادی همچون کراکفورد و اِیک بر مشــکلات امنیتی جاوا اسکریپت اذعــان دارنــد و در عین حال تأکید دارند که ایــن زبان نوآوری‌های قابــل توجهی هم دارد از جمله ســازگاری در مرورگرهای مختلف و پویایی. پیتون جونز معتقد است پژوهش درباره زبان‌های برنامه‌سازی ضعیف اســت چون ســؤال‌های مشــکلی برای پاســخ دادن وجود دارد. گای اســتیل اعتقاد دارد که در گذشــته زبان‌های برنامه‌سازی همچون پاســکال به‌طور کامل طراحی می‌شــده ولی امروزه به‌دلیل بزرگی، زبان‌ها را نمی‌شــود یک‌بــاره طراحی کرد و زبان دوره تکامل را می‌گذراند. طراحی زبان برنامه‌سازی مجموعه‌ایست از موازنه‌ها که طــراح باید با توجه به نیاز برنامه‌نویســان و کاربرد زبان انتخاب و در مورد ویژگی‌های زبان تصمیم‌گیری نماید.

استانداردسازی زبان

برخی همچون کراکفورد معتقدند که زبان نباید خیلی زود استاندارد شــود و باید کمی صبر کرد تا تجربه‌هایــی از کاربردهای واقعی زبان به‌دست آید. زمان‌شناســی، پیروی از استاندارد و کهنگی سه مسئله مهم در استانداردسازی زبان هستند.

دانش ریاضی

تقریبــاً اکثریت اذعان دارند که داشــتن دانش عمیــق ریاضی برای برنامه‌نویسان ضروری نیســت؛ ولی آگاهی نسبی از ریاضی گسسته، آمار و احتمال و منطق ضروری و مفید است. نیاز به بنیاد قوی ریاضی مانع می‌شــود که اکثر برنامه‌ســازان بتوانند کتاب هنر برنامه‌نویسی رایانه کنوث را کامل بخوانند و درک نمایند.

نظام مهندسی، الگوها و چارچوب‌ها

بعضی همچون زاوینســکی به‌خاطر مســایل رقابت در بازار، اعتقادی به اســتفاده از الگوها و چارچوب‌های طراحــی ندارند. اِیک الگوهای طراحی را چندان ســودمند نمی‌داند. ولی بــلاک آگاهی از الگوهای طراحی را بســیار مفید می‌داند؛ شــاید چون مــدرک دکتری او در مهندسی کامپیوتر، نگاه آکادمیک بیشتری به او بخشیده است. امروزه در تولید بسیاری از کتاب‌خانه‌ها و چارچوب‌های نرم‌افزاری از الگوهای طراحی استفاده شده است که آگاهی از آن‌ها به درک سریعتر کمک می‌کند. شاید یکی از هدف‌های الگوهای طراحی، ساختاردهی به کد برنامه‌سازان (مبتدی) و افزایش خوانایی و قابلیت بازبه‌کارگیری است.

آموزش و یادگیری زبان و برنامه‌نویسی

برخــی همچون زاوینســکی اعتقــاد دارند به‌جای قوانیــن گرامری زبان، بایــد طرز عمومی برنامه‌نویســی را آمــوزش داد. تقریباً همه مصاحبه‌شوندگان تأکید دارند که یکی از بهترین روش‌های یادگیری و مهــارت یافتن در برنامه‌نویســی، خواندن کد دیگران اســت، البته کد خوب از برنامه‌نویســان خوب. بلاک معتقد اســت برای یادگیری برنامه‌نویســی نباید از زبانی مثل C شــروع کرد. آرمسترانگ معتقد است که برای درک کامل یک زبان بهتر است مترجمی برای آن زبان بنویسید. گای اســتیل خواندن کد TeX نوشته کنوث را بسیار مفید می‌داند چون خوش نوشتار است و خوب اشکال‌زدایی شده است. آلن هــم خواندن برنامه‌های خوب زبان جدیــد را برای یادگیری آن مؤثر می‌دانــد. یکی از دلایل طراحی زبان‌های جدیــد بر مبنای زبان‌های گذشته، انتقال تجربه‌ها و سهولت آموزش زبان است.

یادگیری جزئیات سیستم‌ها

از صحبت‌هــای افراد می‌توان نتیجه گرفت کــه حتی امروزه آگاهی از جزئیات ســطح پایین سخت‌افزاری و نرم‌افزاری برای برنامه‌نویسان

مفید اســت؛ هر چند که دیگر ۹۹ درصد افراد درگیر چنین جزئیاتی نمی‌شــوند. امروزه برنامه‌نویســی روی لایه‌های بالایی از انتزاع انجام می‌شــود و کســی مجبور نیست سیســتم عامل، ویراستار یا مترجم بنویسد.

تضمین کیفیت، اشکال‌زدایی و وارسی صحت برنامه

تقریباً همه مصاحبه‌شوندگان در عمده فعالیت‌های اشکال‌زدایی خود از دســتور چاپ اســتفاده کرده‌اند. برخی هم از دستور اظهار [11] برای بررســی مقادیر ثابــت [12] در متن برنامه بهره برده‌انــد. عده‌ای هم از امکانات محیط برنامه‌نویسی برای ردیابی کد [13] و گام‌زنی [14] استفاده کرده‌اند. برخی هم ابزارهایی همچون Valgrind و Strace و FindBugs را برای یافتن نوع خاصی از خطاها ســودمند می‌دانند. برنامه‌نویسان جاوا اســکریپت مثل کراکفورد هم از ابزارهای اشکال‌زدایی مرورگرها مثل فایرباگ استفاده کرده‌اند.

تقریبــاً همگــی معتقدند که وارســی صحت برنامــه از طریق اثبات صوری عملی نیســت و اگر هم بخواهیم استفاده کنیم، صرفاً باید در کدهای کوچک و نرم‌افزارهایی که در کاربردهای حساس به‌کار گرفته می‌شوند از اثبات صوری اســتفاده شود. از نظر گای استیل و کنوث، اثبات صحت هم ممکن اســت خطا داشــته باشــد و با اثبات صحت نمی‌تــوان ثابت کرد که برنامه بی‌خطاســت. اما چون اثبات صحت با روش دیگــر و ابزار دیگری صورت می‌گیرد، احتمال وجود خطای آن کمتر از خطای کد اســت. دویچ نیــز اظهار می‌دارد که صوری کردن بسیاری از خصوصیت‌های نرم‌افزار فوق‌العاده مشکل است.

افرادی چون اِیک بیان می‌کنند که برای اشــکال‌زدایی پژوهش‌های کافی صورت نگرفته اســت. این حرف او را می‌توان کاملاً قابل قبول دانســت وقتی که نتایج مقاله اُورســو [۴] را مطالعه کنیم. این مقاله پس از مطالعه‌های موردی به این نتیجه رســیده اســت که با وجود مطالعه‌های بســیار، فنون موجود هنوز در دســت برنامه‌نویســان در موقعیت‌های عملی سودمند نیستند.

تقریبــاً همه مصاحبه‌شــوندگان از مرور کد [15] به‌عنوان روشــی برای تضمیــن کیفیت و صحت کد اســتفاده کرده‌انــد. برخی چون بلاک عاشــق تحلیل‌های ایستا هســتند چون دســته‌ای از خطاها به‌طور خودکار حذف می‌شــوند. به‌نظــر پیتون جونز «هر کجا نوع ایســتا برازنده بود حتماً از آن اســتفاده کنید زیرا مزایای فوق‌العاده‌ای برای نگهداری نرم‌افزار دارد.» نورویگ و تامپسون استفاده از آزمون واحدی [16] و آزمون پس‌نمایی [17] را بــرای تضمین کیفیت مهم می‌دانند. بنابر اظهــارات نورویگ، در گوگل یکی از گام‌های مهم در تضمین کیفیت کد، مرور کد توسط فردی دیگر است.

کتاب‌های مورد مطالعه و توصیه شده

بجز معــدودی همچون زاوینســکی اکثراً کتاب «هنر برنامه‌ســازی رایانه[18]» اثر دونالد کنوث را به‌طور کلی یا جزئی خوانده‌اند و معتقدند کتابی است فاخر که حالت مرجع دارد و در موقعیت‌های لازم باید به آن رجوع کرد. کتاب «الگوهای طراحی[19]» را برخی توصیه می‌کنند و برخی مضر می‌دانند. عده‌ای هم همچون بلاک عقیده دارند کتاب‌های «نفر ماه افســانه‌ای[20]» و «عناصر سبک (نویسندگی[21])» را باید خواند بخصوص که دومی کتاب برنامه‌نویسی نیست ولی بخشی از مهندسی نرم‌افزار نویســندگی اســت. اغلــب توصیه می‌کنند بایــد کتابی در الگوریتم و ساختمان داده‌ها خوانده شود. نورویگ خواندن یک کتاب اشــکال‌زدایی و یک کتاب در نکات و مهارت‌های کدنویسی همچون «کد کامل[22]» را ضروری می‌داند. پیتون جونز هم کتاب‌هایی همچون «مرواریدهای برنامه‌نویسی[23]» جان بنتلی که درباره زیبایی‌های درونی کد است، «ســاختمان داده‌های تابعی محض[24]» از کریس اوکازاکی، «ســاختار و تفســیر برنامه‌های کامپیوتر[25]» از ابلســون و سوزمان و «ترجمه از طریق ادامه‌دهی[26]» از اندرو اپل را توصیه می‌کند.

بهینه‌سازی کد

اغلب مصاحبه‌شــوندگان اعتقاد دارند که بهینه‌ســازی کد توســط برنامه‌نویس بخصوص امروزه اهمیت کمی دارد و خوانایی بسیار مهم‌تر اســت. اگر بخشی از کد قرار اســت میلیون‌ها بار اجرا شود و نرم‌افزار در کاربردهای بیدرنگ اســتفاده می‌شود، آن‌گاه کارایی و بهینه‌سازی اهمیت پیدا می‌کند. تامپسون معتقد است بهینه‌سازی وقت‌گیر است و کد را پیچیده می‌سازد.

پاک‌سازی و بازسازی کد

افرادی همچــون کراکفورد توصیه می‌کنند بعد از هر شــش چرخه نرم‌افزار باید آن‌را مورد پاکســازی اساســی قرار داد. سال‌هاست که فنونی همچون بازســازی کد[27] در محیط‌های توسعه یکپارچه[28] ادغام شده‌اند تا به خوانایی کد و ساختاردهی بهتر آن کمک کنند.

11- Assertion
12- Invariant
13- Trace
14- Step through code
15- Code Review
16- Unit Testing
17- Regression Testing

18- The Art of Computer Programming
19- Design Patterns
20- Mythical Man-Month
21- Elements of Style
22- Code Complete
23- Programming Pearls
24- Purely Functional Data Structures
25- Structure and Interpretation of Computer Programs
26- Compilation with Continuation
27- Refactoring
28- Integrated Development Environment (IDE)

استفاده از برنامه‌نویسی ادیبانه کنوث

اغلب مصاحبه‌شوندگان از برنامه‌نویسی ادیبانه [۲۹] کنوث اطلاع دارند و عده‌ای ضمن تمجید آن، در کدنویسی هم استفاده کرده‌اند. معدودی همچون تامپسون هم ضمن تحسین آن معتقدند در عمل قابل استفاده نیست. گای استیل برنامه‌نویسی ادیبانه را در کار خودش مؤثر می‌داند.

شناخت برنامه‌نویس خوب برای استخدام

تقریباً همه بیان کرده‌اند که موقع مصاحبه برای استخدام برنامه‌نویس طرح سوال‌ها و مسئله‌های معما گونه که در شرکت‌هایی مثل گوگل و مایکروسافت رایج است، سودمند نیست. آرمسترانگ اعتقاد دارد برخی از برنامه‌نویسان خوب در حل معماها کند هستند. بیشتر افراد علاقه‌مندند میزان تسلط فرد بر ساختمان داده‌ها و الگوریتم‌ها را بدانند و از او بخواهند از چگونگی حل مسئله، جزئیات کارهای عملی و بهترین پروژه‌ای که انجام داده صحبت کند. برنامه‌نویس خوب باید در موقعیت‌های برنامه‌نویسی بتواند تصمیم‌های منطقی بگیرد و بلد باشد خودش را با محیط و ابزار جدید وفق دهد و کار با آن‌را بیاموزد. تسلط بر یک زبان خاص دلالت بر خوبی برنامه‌نویس نیست. نورویگ معتقد است فردی که می‌خواهد جذب صنعت شود باید شیوه کارِ گروهی را بلد باشد. این نیازمندی و نیز اشکال‌زدایی را در زمره مهارت‌هایی می‌داند که در دانشگاه آموزش داده نمی‌شوند. کاسل برای استخدام به دنبال فردی بوده است که جست‌وجوگر، دقیق و کنجکاو باشد و رویکرد منطقی به حل مسایل داشته باشد.

ضرورت تحصیلات آکادمیک برای برنامه‌نویسی

نظر اِیک اینست که همه نیاز به دکتری رایانه ندارند و او از تجربه حضور در دره سیلیکان فکر می‌کند که معامله اقتصادی سودآوری نیست. به‌نظر می‌رسد امروزه که برنامه‌نویسی در سطوح انتزاعی گوناگونی صورت می‌گیرد، داشتن تحصیلات کارشناسی برای برنامه‌نویسی در سطح رابط گرافیکی کاربر و طراحی وب کافی باشد؛ اما برای برنامه‌نویسی در سیستم‌های مقیاس‌پذیر که محاسبات پیچیده‌ای دارند و ملاحظات فنی بسیاری لازم است، تحصیلات آکادمیک در مقاطع بالاتر می‌تواند مفید باشد.

خطاهای داخل کد

اکثر برنامه‌نویسان بیان کرده‌اند که خطاهای موجود در برنامه‌های هم‌روند، چندریسمانی و موازی در زمره دشوارترین خطاها برای اشکال‌زدایی و نوشتن آزمایه هستند. همچنین اِیک از خطاهای محل مشتری بیان می‌کند که مدت زمان زیادی باقی نمی‌مانند ولی یافتن آن‌ها دشوار است. برخی از خطاها هم در مواقع نادر رخ می‌دهند که کشف آن‌ها معمولاً ساده نیست. به‌نظر گای استیل، زبان از نوع خاصی از خطاها جلوگیری می‌کند.

طرز طراحی و کدنویسی

بلاک توصیه می‌کند کدی که از واسط برنامه کاربردی [۳۰] استفاده می‌کند را پیش از خود واسط بنویسید. افرادی همچون بلاک با روش برنامه‌نویسی دونفره[۳۱] بسیار کار کرده‌اند و معتقدند ضمن حفظ استقلال، خوب است برنامه‌نویسان یک تیم از کد همدیگر اطلاع داشته باشند. در مقابل اینگالس برنامه‌نویسی دونفره را نمی‌پسندد و تجربه هم نکرده است. برخی همچون نورویگ ابزارهای مدل‌سازی همچون UML را چندان دوست ندارند. تامپسون و دویچ بر طراحی نرم‌افزار و طراحی ساختمان داده‌های مناسب پیش از شروع کدنویسی تأکید دارند و دویچ اندازه‌گیری و مقیاس‌بندی را بخصوص در مواجهه با نرم‌افزارهای بزرگ مهم می‌داند. کاسل بر طراحی برنامه به‌طور آزمایش‌پذیر تأکید دارد.

کلام آخر با چند جمله کوتاه و خواندنی

پیتر دویچ: «سریع، ارزان، خوب: دوتایش را انتخاب کن.»

دونالد کنوث: «برنامه‌نویسی شبیه اعتقادات مذهبی است.»

پیتون جونز: «ویژگی هور: به‌جای نداشتن خطاهای واضح، واضح است که خطایی ندارد.»

جاشوا بلاک: «بهینه‌سازی کد صحیح آسان‌تر از تصحیح کد بهینه‌سازی شده است.»

داگلاس کراکفورد: «هر چه زبان موفق‌تر باشد، هزینه تغییر در آن بیشتر است.»

ادگار دایکسترا: «اگر نمی‌توانید به زبان مادریتان بنویسید، برنامه‌نویسی را رها کنید.»

مراجع

[1] Seibel, Peter. Coders at work: Reflections on the craft of programming. Apress, 2009.

[2] Sebesta, Robert W. Concepts of programming languages, 11th Ed., Pearson , 2016.

[3] Scott, Michael L. Programming language pragmatics, Morgan Kaufmann, 2009.

[4] Parnin, Chris, and Alessandro Orso. "Are automated debugging techniques actually helping programmers?." In Proceedings of the 2011 international symposium on software testing and analysis, pp. 199-209. ACM, 2011.

29- Literate Programming

30- Application Program Interface (API)

31- Pair programming

آموزش مهارت‌های پایه و ترویج حرفه‌ای‌گری
راه‌حلی برای پیشگیری از افزایش دانش‌آموختگان بی پیشه

سید ابراهیم ابطحی
استادیار دانشکده مهندسی کامپیوتر- دانشگاه صنعتی شریف
پست الکترونیکی: abtahi@sharif.edu

پیشگفتار

بی‌پیشـگی دانش‌آموختـگان دانشـگاهی در عین نیاز به آن‌هـا، که اینک در بخش‌هائی با آن مواجهیم، نیاز به آسیب‌شناسـی دارد. این مشـکل دارای اثرات اجتماعی و فردی گسـترده اسـت که چاره‌اندیشـی برای مقابله با آن را ناگزیر می‌سازد. در یک بررسی اجمالی، وجهی از این دشواری را به کاهش توان نهادهای آموزشگر در تبدیل سرمایه‌های انسانی به سرمایه‌های فکری، می‌توان نسبت داد و از علل آن، به گسـترش کمّی بی‌رویّه ظرفیت پذیرش دانشجو، اشاره کرد. اما در یک علت کاوی ریشه‌یابانه، می‌توان به کمبودهائی با پیشینه تاریخی طولانی‌تر، توجه نمود. از جمله فقدان یا قلت دانش و مهارت‌های پایه دانشجویان، که آن‌ها را در حین تحصیل، با کاهش اثربخشی یادگیری زنجیره‌های درسی و کم تاثیری آن‌هـا، در افزایش دانش و مهارت‌های موضوعـی، مواجه می‌کند. در پایان دوره کارشناسـی، فقدان مهارت‌های حرفه‌ای هم، یکی از عوامل موثر در بی‌پیشـگی اسـت. به این مسـئله و وجوه آن، قبلا در نوشـته‌های متعـدد، در قالب دیدگاه، پرداخته‌ایم. در این شـماره در قالب معرفی کتاب، می‌خواهیم به استاد پیشگامی که سال‌هاسـت آینده نگرانه با نوشته‌ها، مقالات و کتب خود و تدریس آن‌ها در دانشگاه، در صدد کمک به حل این دشواری است، بپردازیم و کتاب‌های ایشان را که پیش از این یکی آن‌ها را به مناسبتی دیگر معرفی نمودیم، را بازخوانی کنیم.

دکتر حسین معماریان استاد تمام دانشکده فنی دانشگاه تهران و در حال حاضر ریاسـت کرسی آموزش مهندسـی یونسـکو در ایران، که در اینک عضو هیئت مدیره و رئیس کمیته پژوهش انجمن آموزش مهندسـی هسـتند، سال‌هاسـت به انجام اقدامات موثر تالیفی و تدریسـی در این حوزه اشـتغال دارند.سه کتاب دانشـگاهی ایشـان با عناوین «حرفه مهندسـی»، «نوآوری در آموزش مهندسی» و «طراحی مهندسـی»، که در این شـماره به معرفی آن‌هـا خواهیم پرداخت، در موضوعات مطروحه از اولین کتاب‌های تالیفی دانشـگاهی به زبان فارسی هستند که از این بابت هم باید قدردان این استاد گرانقدر بود.

عنوان کتاب: حرفه مهندسی
نویسنده: حسین معماریان.
ناشر: انتشارات دانشگاه تهران.
زمان نشر: چاپ دوم، ۱۳۹۱.
تعداد صفحات: ۵۳۴ برگ مصور.
شمارگان: ۱۰۰۰ نسخه.
شابک: ۹۷۸-۹۶۴-۰۳-۵۸۶۱-۰
قیمت: ۱۱۰۰۰ تومان.

مقدمه

در پیشـگفتار کتاب، مولف نوشته اسـت: «آموزش دانشگاهی به ما فنون، روابط و فرآیندهای یک رشته‌های مهندسی را با جزئیات بسیار می‌آموزد، ولی کمتر آموزشی در مورد نحوه به کارگیری آن‌ها در دنیای واقعی، ارائه می‌کند. آنچه در این کتاب فراهم آمده، مجموعه‌ای اسـت که مهندسان به‌طور معمول آن‌ها را به صورت غریزی، یا با آزمون و خطا پس از سال‌ها کار حرفه‌ای در صنعت، فرا می‌گیرند. کتاب حاضر سعی دارد که مجموعه‌ای از این مهارت‌ها را در اختیار دانشجویان قرار دهد».

محتوای کتاب

کتـــاب دارای پیشـــگفتار مولف و نوزده فصل در پنـــج بخش و یازده پیوســـت، مراجع کتاب، واژه نامه و فهرست راهنما از جمله فهرست حاشیه‌ها است. عناوین بخش‌ها و فصل‌های کتاب به شرح زیر است:

بخش یک: آموزش مهندسی

فصل ۱: مهندس و مهندسی.
فصل ۲: آموزش مهندسی در ایران.
فصل ۳: دانشجوی مهندسی.
فصل ۴: زبان مهندسی.

بخش دوم: ارتباطات مهندسی

فصل ۵: ارتباط شفاهی.
فصل ۶: ارتباط الکترونیکی.
فصل ۷: ارتباط نوشتاری.
فصل ۸: ارتباط تصویری.

بخش سوم: بازار کار مهندسی

فصل ۹: در جست و جوی کار.
فصل ۱۰: مهندس حرفه‌ای.
فصل ۱۱: اخلاق مهندسی.

بخش چهارم: پژوهش در مهندسی

فصل ۱۲: گردآوری داده‌ها.
فصل ۱۳: نگارش مقاله‌های پژوهشی.
فصل ۱۴: مالکیت فکری.

بخش پنجم: عملیات مهندسی

فصل ۱۵: برآورد مهندسی.
فصل ۱۶: طراحی مهندسی.
فصل ۱۷: برنامه ریزی مهندسی.
فصل ۱۸: ایمنی در مهندسی.
فصل ۱۹: مدیریت ریسک.

عنوان کتاب: نوآوری در آموزش مهندسی
نویسنده: حسین معماریان.
ناشر: انتشارات دانشگاه تهران.
زمان نشر: چاپ دوم، ۱۳۹۲.
تعداد صفحات: ۴۳۸ برگ مصور.
شمارگان: ۱۰۰۰ نسخه.
شابک: ۰-۵۸۶۱-۰۳-۹۶۴-۹۷۸
قیمت: ۱۶۰۰۰ تومان.

مقدمه

در پیشـــگفتار کتاب، مولف نوشـــته اســـت: «توانائی حرفه‌ای را می‌توان ارتباط پیچیده بین ســـه مقولـــه دانش، مهارت و نگرش به حســـاب آورد. به این منظور، یک دوره آموزشی مهندسی باید بتواند دانش‌آموختگانی تربیت کند که ضمن داشتن دانش کافی از مبانی علوم و مهندسی، دارای مهارت و توانائی در ارتباطات، کار گروهی و خودآموزی باشـــند و نگرش درستی نسبت به محتوای اجتماعی، اقتصادی و زیســـت محیطی مهندســـی داشته باشند... هـــدف این کتاب ارائه روش‌هائی بـــرای بالابردن کیفیت آموزش مهندسی است».

محتوای کتاب

کتاب دارای فهرست مطالب، پیشگفتار مولف، چهارده فصل در چهار بخش، منابع، دو واژه‌نامه و فهرســـت راهنما است. عناوین بخش‌ها و فصل‌های کتاب به شرح زیر است:

بخش اول: تدارک آموزش مهندسی

فصل ۱: مهندسی و آموزش.
فصل ۲: طراحی تدریس مهندسی.
فصل ۳: تدارک هدف‌ها و دستاوردها.

بخش دو: تدریس و یادگیری مهندسی

فصل ۴: سخنرانی در کلاس.
فصل ۵: آموزش دانشجو محور.
فصل ۶: یادگیری فعال.
فصل ۷: ارزیابی یادگیری.

بخش سوم: تضمین کیفیت آموزش مهندسی

فصل ۸: ارزشیابی در جهان.
فصل ۹: ارزشیابی در ایران.
فصل ۱۰: ساز و کار ارزیابی درونی.

بخش چهار: برنامه ریزی آموزشی

فصل ۱۱: نگرشی تازه در آموزش مهندسی.
فصل ۱۲: طراحی درس جدید.
فصل ۱۳: طراحی برنامه کارشناسی جدید.
فصل ۱۴: کاستی‌های آموزش مهندسی ایران.

عنوان کتاب: طراحی مهندســـی: خودآموز، هدف دار و پروژه محور
نویسنده: حسین معماریان.
ناشر: انتشارات دانشگاه تهران.
زمان نشر: چاپ اول، ۱۳۹۲.
تعداد صفحات: ۵۳۴ برگ مصور.
شمارگان: ۱۰۰۰ نسخه.
شابک: ۰-۵۸۶۱-۰۳-۹۶۴-۹۷۸
قیمت: ۲۴۰۰۰ تومان.

مقدمه

در پیشـــگفتار کتاب، مولف نوشته است: «طراحی مهندسی یک فراینـــد معیوب و با انتهای باز اســـت، و بر خلاف دیگر مســـائل مهندســـی یک راه‌حل واحد نداشـــته و راه‌حل‌های متصور آن را نیز نمی‌توان با یک الگوریتم یا روش ریاضی خاص، تعیین کرد. بـــه عنوان مثال، با الگوریتم ها و روابط ریاضی نمی‌توان نیازهای سودبران اصلی طراحی، یعنی کارفرما، مصرف‌کنندگان و اجتماع را تعیین کرد. برای بررســـی فعالیت‌های دارای ساختار معیوب و کمک بـــه تصمیم‌گیری و انتخاب بهینـــه از میان گزینه‌های محتمل، مهندســـان فرایند طراحی را ســـامان داده و روش‌ها و ابزارهایی را برای انجام بایســـته آن، به کار می‌برند. تاکید اصلی کتاب حاضر بر طراحی مفهومی یا بخشی از فرایند طراحی است کـــه منجر به یافتن گزینه‌های طراحی می‌شـــود. کتاب طراحی مهندسی به صورت خودآموز، هدفدار و پروژه محور طراحی شده است».

محتوای کتاب

کتاب دارای فهرســـت مطالب، پیشـــگفتار مولف، چهارده فصل در دو بخش، منابع و مراجع، دو واژه نامه و فهرســـتی راهنما است. عناوین بخش‌ها و فصل‌های کتاب به شرح زیر است:

بخش یک: طراحی مهندسی چیست؟

فصل ۱: الفبای طراحی مهندسی.
فصل ۲: فرایند طراحی مهندسی.
فصل ۳: روش‌ها و ابزارهای طراحی.
فصل ۴: جایگاه خلاقیت در طراحی مهندسی.
فصل ۵: طراحی مهندسی و کار تیمی.

بخش دوم: طراحی مهندسی چگونه انجام می شود

فصل ۶: تعریف مسئله طراحی.
فصل ۷: عملکردها و مشخصات طراحی.
فصل ۸: یافتن پاسخ مسئله طراحی.
فصل ۹: گزارش دستاورد طراحی.
فصل ۱۰: مدیریت فرایند طراحی.
فصل ۱۱: ویژگی‌های یک طراحی خوب.
فصل ۱۲: نقش اخلاق در طراحی مهندسی.
فصل ۱۳: حل نوآورانه مسئله طراحی (تریز).
فصل ۱۴: آموزش طراحی مهندسی.

مطالعه ســـه کتاب فوق برای مدرسان و دانشجویان غالب رشته‌های مهندسی ثمر بخش و قابل توصیه است و می‌تواند به شکل چشمگیری بر شکل و محتوای آموزش مدرسان و یادگیری دانشجویان اثر مثبت بگذارد.

سؤالات هجدهمین مسابقهٔ منطقه‌ای برنامه‌نویسی دانشجوئی ACM

ترجمهٔ: کیان اهرابیان
دانشجوی علوم کامپیوتر، دانشگاه تهران
پست الکترونیکی: kahrabian@ut.ac.ir

پیش گفتار

هجدهمین مسابقهٔ منطقه‌ای برنامه‌نویسی دانشجوئی انجمن ماشین‌های رایانشی (ACM) در تاریخ سوم دی ماه ۹۵ در دانشکدهٔ مهندسی کامپیوتر دانشگاه صنعتی شریف برگزار شد.

در این مسابقه ۸۲ تیم سه نفره از دانشگاه‌های مختلف سراسر کشور شرکت کرده بودند و باید در مدّت ۵ ساعت، ۱۲ مسئله را حل می‌کردند.

گزارشی از این مسابقه در شمارهٔ قبل به چاپ رسید. در این شماره، سؤالات این مسابقه به اطلاع علاقه‌مندان می‌رسد.

* * *

مسئله ۱- مالیات

مقدار مالیات کسر شده از درآمد هر فرد به میزان درآمد او بستگی دارد. برای درآمد کمتر یا مساوی ۱،۰۰۰،۰۰۰ اشلوب مالیاتی از آن کم نمی‌شود. برای درآمد بیشتر از ۱،۰۰۰،۰۰۰ و کمتر یا مساوی ۵،۰۰۰،۰۰۰ مقدار مالیات ۱۰٪ کل درآمد می‌باشد و برای درآمد بیشتر از ۵،۰۰۰،۰۰۰ مقدار مالیات ۲۰٪ می‌باشد. برنامه‌ای بنویسید که مقدار درآمد هر فرد پس از کسر مالیات را به‌دست آورد.

ورودی

تعدادی خط که در هر خط یک عدد مثبت قرار دارد که مضرب ۱۰۰۰ و کمتر از ۱۰،۰۰۰،۰۰۰ می‌باشد که درآمد فرد قبل از کسر مالیات است. برنامه با دریافت صفر تمام می‌شود.

خروجی

برای هر فرد در یک خط یک عدد که درآمد او پس از کسر مالیات می‌باشد را چاپ کند.

مسئله ۲- کلیدساز

حسن یک کلید ساز خوشحال است. هر مشتری او کلیدی دارد که روی آن چند بریدگی با عمق‌های مختلف می‌باشد. حسن برای ساختن کپی از یک کلید باید روی یک کلید صاف به همان تعداد بریدگی و با عمق‌های یکسان و به همان ترتیب ایجاد کند. او در روزهای ابتدایی کارش تعداد زیادی کلید صاف استفاده کرد که با کلید مشتری‌ها مطابقت نداشت و نتوانست آن‌ها را بفروشد. به همین دلیل آن‌ها را در جعبه‌ای نگه داشت تا دوباره از آن‌ها استفاده کند.
هر مشتری که برای ساخت کپی از کلیدش می‌آید حسن جعبه را بررسی می‌کند و کلیدهایی که قابلیت تبدیل به کلید مشتری را دارند می‌شمارد. کلیدی قابلیت تبدیل به کلید دیگری را دارد که عمق تمام بریدگی‌های آن کمتر یا مساوی عمق‌های کلید دیگر به همان ترتیب باشد. محل و تعداد بریدگی تمام کلیدها یکسان است.

ورودی

چندین آزمایه در ورودی قرار دارد که در هر کدام از آن‌ها در خط اول ۲ عدد صحیح m که بیانگر تعداد بریدگی‌های کلیدها ($1\le m\le 10$) و n تعداد کلیدهای درون جعبه با همان تعداد بریدگی می‌باشد ($1\le m\le 100$). در خط بعد m عدد صحیح، عمق بریدگی‌های کلیدهای مشتری‌ها داده می‌شود. هر یک از n خط بعدی شامل m عدد صحیح، عمق بریدگی‌های هر کلید داخل جعبه است. ورودی با دو صفر تمام می‌شود.

خروجی

برای هر آزمایه تعداد کلیدهایی را که می‌توان آن‌ها را به کلید مشتری تبدیل کرد در یک خط چاپ کنید.

مسئله ۳- لوگوی المپیاد بین‌المللی انفورماتیک

میزبان المپیاد بین‌المللی انفورماتیک ۲۰۱۷ ایران می‌باشد. تیم برگزاری برای طراحی لوگوی مسابقه اعلان عمومی کرد. با توجه به فعالیت زیاد نسل جوان ایران در هر واقعهٔ ملی تعداد بسیاری لوگو در مدت کوتاهی دریافت شد. در مرحلهٔ اول تعدادی طراح گرافیک حرفه‌ای برخی از لوگوها را که از نظر هنری نسبت به بقیه بهترین بودند انتخاب کردند تا در مرحلهٔ دوم بهترین در بین آن‌ها را انتخاب کنند.
حال لوگوهای انتخاب شده به کمیته برگزاری برای رأی‌گیری ارائه شده است. اما رأی‌گیری آن‌ها کمی عجیب است: بدین صورت که هر عضو حداکثر به ۳ لوگو می‌تواند رأی بدهد. که انتخاب اول و دوم و سوم هر فرد به ترتیب ۳و۲و۱ امتیاز دریافت می‌کند. امتیاز هر لوگو مجموع امتیازهای آن از تمامی اعضا می‌باشد. لوگویی برنده می‌باشد که بیشترین امتیاز را داشته باشد. در صورت تساوی لوگویی که بیشترین رای ۳ امتیازی را دارد و در صورت برابری رأی‌های ۳ امتیازی لوگویی که رای ۲ امتیازی بیشتری دارد و در صورت برابری رأی‌های ۲ امتیازی همهٔ آن لوگو ها برنده می‌باشند.

ورودی

شامل تعدادی آزمایه می‌باشد. در خط اول هر آزمایه یک عدد صحیح n نشانگر تعداد اعضای کمیته می‌باشد($1\le n\le 100$). سپس برای هر عضو کمیته یک خط که در ابتدای آن یک عدد صحیح diکه تعداد لوگوهای انتخاب شده توسط iامین رأی‌دهنده می‌باشد ($1\le di\le 3$) و در ادامه شمارهٔ لوگوهایی که به آن‌ها رأی می‌دهد به ترتیب داده می‌شود. در صورت صفر بودن عدد اول برنامه تمام می‌شود.

خروجی

برای هر آزمایه شمارهٔ لوگو یا لوگوهای برنده را به ترتیب صعودی در یک خط چاپ کنید.

مسئله ۴- رتبه‌بندی میکرو آر.ان.ای

احلام یک دانشجوی علوم کامپیوتر است که پایان نامهٔ دکتری او یک پروژهٔ بیوانفورماتیک است در مورد انواعی از مولکول‌های خاص که در سلول‌ها پیدا می‌شوند (میکرو آر.ان.ای ها). او در پروژه‌اش به دنبال میکرو آر.ان. ای‌هایی است که به فاکتور خاصی از سلامتی انسان‌ها مربوط می‌باشند.
او تعدادی الگوریتم طراحی کرده است که تمام میکرو آر.ان.ای ها را از یک دیدگاه خاص رتبه‌بندی می‌کنند. هر الگوریتم یک جایگشت از میکرو آر.ان.ای ها به ما می‌دهد که اولین میکرو آر.ان.ای بیشترین رتبه و آخرین آن‌ها کمترین رتبه را دارد.
احلام به دنبال یک رتبه‌بندی توافقی است. به یک رتبه‌بندی توافقی می‌گوییم هرگاه رتبهٔ میکرو آر.ان.ای i از j کمتر باشد در رتبه‌بندی حداقل نصف الگوریتم‌ها از آن کمتر باشد. برنامه‌ای بنویسید که به او کمک کند یک رتبه‌بندی توافقی پیدا کند.

ورودی

تعدادی آزمایه داده می‌شود که در هرکدام در خط اول ۲ عدد صحیح n ($1\le n\le 1000$) و ($1\le k\le 200$) که به ترتیب بیانگر تعداد میکرو آر.ان. ای ها و تعداد الگوریتم‌ها می‌باشند. در ادامه به تعداد الگوریتم‌ها در هر خط یک جایگشت از میکرو آر.ان.ای‌ها می‌دهد که رتبه‌بندی آن‌ها طبق آن الگوریتم می‌باشد. در صورتی که ۲ عدد خط اول صفر باشند برنامه تمام می‌شود.

خروجی

برای هر آزمایه یک رتبه‌بندی توافقی به‌دست آورید. در صورت وجود چند رتبه‌بندی توافقی جوابی که از نظر ترتیب لغت‌نویسی اول می‌باشد را چاپ کنید. در صورتی که جوابی وجود نداشت عبارت "No solution" را چاپ کنید.

مسئله ۵- نوار موبیوس

نوار موبیوس از نوار کاغذی بلندی درست شده، به این صورت که یک سر نوار را ۱۸۰ درجه بچرخانیم و به سر دیگر وصل کنیم.
به جای این‌که سر نوار را ۱۸۰ درجه بچرخانیم، می‌توانیم آن را مضربی از ۱۸۰ درجه بچرخانیم. نوع نوار عدد صحیح و مثبتی است که برابر تعداد چرخش‌های ۱۸۰ درجه نوار است.
در هر حرکت می‌توانیم از یک نقطه روی نوار به فاصله $\frac{۱}{۳}$ از یکی از لبه‌های آن شروع به بریدن کنیم و یک دور کامل بزنیم. بعد از بریدن، نوار به چند تا نوار دیگر تبدیل شود که می‌توان عمل بریدن را روی هر یک از آن‌ها جداگانه انجام داد. (ممکن است دو نوار جدید درون همدیگر باشند که باز هم دو نوار جدا در نظر می‌گیریمشان.)
حال دو مجموعه از نوارها داده شده‌است، می‌خواهیم ببینیم آیا می‌توان با تعدادی عملیات بریدن رو هر مجموعه، به دو مجموعهٔ جدید برسیم که تعداد نوارها از هر نوع، در هر دو مجموعه برابر باشد؟

ورودی

در هر ورودی تعدادی آزمایه آمده که برای هر کدام در ابتدا دو عدد a و b آمده که برابر تعداد نوارهای هر دسته است. در دو خط بعدی به ترتیب a و b عدد آمده که هر کدام نشان دهندهٔ نوع نوار است و حداکثر ۱۰۰ می‌تواند باشد. ورودی با یک خط ۱- ۱- به اتمام می‌رسد.

خروجی

برای هر آزمایه، در یک خط اگر می‌توان به دو مجموعهٔ یکسان رسید Y و در غیر این صورت N چاپ کنید.

مسئله ۶- عبارت جبری

به دلیل برف زیاد علی با تاخیر به کلاس رسید. وقتی او به کلاس رسید با تمرین سختی که معلم ابتدایی او روی تخته نوشته بود برخورد کرد. تمرین یک ترکیب پیچیده از جمع و ضرب و تقسیم روی اعداد مثبت بود. مانند عبارت زیر:

$$۵+\frac{۱}{۱+۱+۲\times\frac{۴\times\frac{۶}{۷}}{۲+۳\times۵}}$$

هر عبارت یا زیرعبارت به‌صورت یک جمع یا ضرب یا تقسیم چند زیر عبارت دیگر است.
تقسیم هر ۲ عبارت به صورت یک خط کسری می‌باشد که صورت مقسوم می‌باشد که بالای خط کسری است و مخرج یا همان مقسوم علیه پایین خط کسری می‌باشد. توجه داشته باشید که اولویت ضرب نیز بیشتر از جمع می‌باشد.
به علی کمک کنید تا حاصل عبارت روی تخته را به صورت یک کسر ساده در آورد.

ورودی

تعدادی آزمایه می‌باشد که در هر کدام در خط اول تعداد خط‌های عبارت به شما داده می‌شود و سپس به همان تعداد خط قسمت‌های عبارت به شما داده می‌شود.
«*» بیانگر ضرب می‌باشد و «+» بیانگر جمع و تعدادی «-» کنار یکدیگر بیانگر یک خط کسری می‌باشند. فاصلهٔ بین اولین رقم صورت و مخرج هر کسر با اولین «-» آن خط یک می‌باشد. در صورت صفر بودن عدد ابتدایی برنامه تمام می‌شود.

خروجی

برای هر آزمایه یک خط که ساده شدهٔ عبارت به‌صورت یک کسر ساده می‌باشد و به‌صورت «مخرج/صورت» می‌باشد چاپ کنید. توجه داشته باشید که صورت و مخرج باید نسبت به هم اول باشند.

مسئله ۷- انتخابات

جناب خان که به تازگی از شغل لبو فروشی میلیاردر شده است برای ریاست جمهوری نامزد شده است. کشور او از یک روش عجیب برای انتخاب رییس جمهور استفاده می‌کند. بدین ترتیب که کشور او از تعدادی استان تشکیل شده است که هر کدام مستقل از بقیه رأی می‌دهند. هر استان با توجه به جمعیتش تعدادی نماینده در هیئت انتخاب کنندگان رییس جمهور دارد که آن‌ها با توجه به تعداد رأی‌های هر نامزد در ایالت خود رأی می‌دهند. در صورت تساوی هر استان یک انتخاب برای مشخص شدن برنده دارد. نامزدی که بیش از نصف آرای هیئت انتخاب کننده رییس جمهور را کسب کند رییس جمهور می‌شود.
برای هر استان احتمال پیروزی جناب خان در آن استان به شما داده می‌شود. احتمال رییس جمهور شدن جناب خان را به‌دست آورید.

ورودی

شامل تعدادی آزمایه است که در هرکدام در خط ابتدایی یک عدد که تعداد استان‌ها می‌باشد($۱۰۰۰\geq n\geq ۱$) و سپس به ازای هر استان یک خط که احتمال پیروزی جناب خان در آن استان با دقت ۴ رقم اعشار ($۱\geq p_i\geq ۰$) و تعداد اعضای آن استان در هیئت می‌باشد داده می‌شود.
تعداد اعضای هیئت عددی فرد و کمتر از ۲۰۰۰ می‌باشد.

خروجی

برای هر آزمایه احتمال رییس جمهور شدن جناب خان را با دقت ۴ رقم اعشار در یک خط چاپ کنید.

مسئله ۸- انفجار در کافه بازار

۸- شما یک مهندس در کافه بازار هستید و تخصص شما شبکه‌های ارتباطی و رفتارهای آن‌ها می‌باشد. همکار شما یک شبکه طراحی کرده است که دارای تعدادی سوییچ می‌باشد که تعدادی پیوند جهتدار بین آن‌ها وجود دارد. هر سوییچ می‌تواند مقدار داده را در

خود نگـه دارد و ۲ حالت دارد: ۱- حالت فرسـتادن: در این حالت سـوییچ داده خود را به تمامی سـوییچ‌هایی که به آن‌ها پیوند دارد می‌فرستد و داده ها از روی سوییچ پاک می‌شوند. ۲- حالت دریافت: سـوییچ تمامی اطلاعاتی که در سوییچ‌هایی که به این سوییچ پیوند دارنـد را دریافت می‌کند و آن‌ها را ذخیـره می‌کند. در نتیجه اندازه داده‌های ذخیره شـده در آن جمع اندازهٔ تمام داده‌های سوییچ‌هایی که از آن‌ها اطلاعات دریافت می‌کند می‌شود.

در ثانیـه صفر تمامی سـوییچ‌ها روی حالت اول می‌باشـند و در هر ثانیه حالت آن‌ها تغییر می‌کند. در ابتدا تمامی سوییچ‌ها بجز سوییچ i داده‌ای ندارند و اندازهٔ دادهٔ این سـوییچ یک بیت می‌باشـد. سوییچ i انفجاری گفته می‌شـود اگر با گذشـت زمان تا بینهایت اندازه دادهٔ ذخیره شده در سوییچ‌ها نیز به بینهایت میل کند.

برنامه‌ای بنویسـید که تعداد سـوییچ‌های انفجاری در یک شبکه را به‌دست آورد.

ورودی

تعدادی آزمایه داده می‌شـود که در هر کدام از آن‌ها در خط ابتدایی ۲ عدد به ترتیب تعداد سوییچ‌های شبکه و تعداد پیوندهای آن می‌باشند. سپس به ازای هر پیوند یک خط که سوییچ ابتدایی و انتهایی آن است داده می‌شود. در صورت صفر بودن ۲ عدد ابتدایی برنامه تمام می‌شود. تعداد سوییچ‌ها و پیوندها کوچکتر یا مساوی ۵۰،۰۰۰ است.

خروجی

برای هر آزمایه یک عدد در یک خط که تعداد سـوییچ‌های انفجاری می‌باشد.

مسئله ۹- کهکشان خطی

دولت کهکشـان «خطی» در نظر دارد تا سیستم حمل و نقل عمومی را بازطراحی کند. کهکشـان خطی که از کهکشان ما، راه شیری، در فاصلهٔ بسیار دوری قرار دارد از ستاره که روی یک خط واقع شده‌اند تشـکیل شده اسـت. با فرض آن‌که ابتدای خط مذکور بر روی نقطه شروع کهکشـان قرار دارد، ستاره i-ام کهکشان در موقعیت x_i قرار دارد. در نتیجه فاصله بین ستاره i-ام و j-ام برابر $|x_i - x_j|$ است.

در این کهکشـان دو نوع حمل و نقل عمومی به نام‌های TRT و SRT وجود دارد.

حمل و نقل سـریع دوربرد یا TRT یک سیستم حمل و نقل پیشرفته اسـت که با استفاده از آن فردی می‌تواند به صورت آنی از یک ستاره در کهکشان خطی به ستاره‌ای دیگر از همان کهکشان منتقل شود. اما به دلیـل محدودیت‌های ابزاری، پایگاه‌های TRT در تنها $2^{n-1}+1$ ستاره قابل نصب هستند.

برای این‌که شـبکه حمل و نقل همبند شـود، یکی از سـتاره‌های شـبکه TRT و تمامی ستاره‌هایی که در این شـبکه نیستند، که در کل $2^{n-1}+1$ سـتاره می‌شـوند، باید از طریق حمل و نقل سریع سـفینه‌ای یا SRT به هم متصل شوند. یک شـبکه استاندارد SRT با m سـتاره یک دور به طول m اسـت که از هر سـتاره دقیقا یک بار عبور می‌کند. جالب توجه اسـت که سفینه‌های فضایی در پروازهای طولانی‌تـر کارآمدتر هسـتند. در نتیجه کارایی یک شـبکه SRT به صـورت کمینه فاصله‌هـای پروازی (فاصله بین هر دو جفت سـتاره متوالی در دور) تعریف می‌شود.

حال دولت کهکشـان خطی به دنبال شبکه‌ای SRT با بیشینه کارایی که بر روی $2^{n-1}+1$ ستاره ساخته می‌شود است.

ورودی

در ورودی تعدادی نمونه آزمایشـی n می‌آیـد. در خط اول هر نمونه آزمایشـی می‌آید که 2^n+1 تعداد ستاره‌های کهکشان خطی است ($1 \le n \le 10$). خط بعد شـامل 2^n+1 عدد صحیح متمایز اسـت که از 10^9 تجاوز نمی‌کنند و نشان‌دهندهٔ موقعیت ستاره‌ها می‌باشند. ورودی با یک خط صفر پایان می‌یابد که نباید پردازش شود.

خروجی

برای هر نمونه آزمایشـی، در یک خط تنها یک عدد صحیح خروجی چاپ کنید که نشـان‌دهندهٔ بیشـینه کارایی در بین تمام شبکه‌های SRT/TRT است.

مسئله ۱۰- رهیاب

پس از پایان تحصیلاتش، باب اسفنجی یک شرکت نرم‌افزاری به اسم «رهیاب تک» تاسیس کرد. رهیاب تک برنامهٔ موبایلی به اسم رهیاب را توسعه می‌دهد که به راننده‌ها در پیدا کردن مسیر در مسافرت‌های بین شهری کمک می‌کند.

باب اسـفنجی فهمید که یکی از سخت‌ترین مسائلی که برنامه‌اش با آن روبروسـت، پیدا کردن «مسیر» برای کسـانی است که جمعه‌ها می‌خواهند از متل قو به تهران برگردند. هر مسـیر از متل قو شـروع می‌شـود و پس از گذر از چند «جاده» و چند شـهر واسطه به تهران ختم می‌شود. هر جاده یک مسیر یک طرفه از یک شهر به شهر دیگر است. جاده ها ظرفیت نامحدود دارند و در آن‌ها ترافیک نمی‌شود اما مشـکل اینجاسـت که هرچقدر از یک جاده ماشین بیشتری بگذرد، جاده سریع‌تر خراب می‌شود. برای همین دولت تصمیم گرفته عوارض را طبق شـلوغ‌ترین جاده‌ای که هر فرد در طول مسـیرش طی کرده بگیرد. به این صورت که ادارهٔ راهداری دولت تعداد خودروهایی که از هر جاده در طول روز می‌گذرد را می‌شـمارد، و در پایان از هر خودرو برای شـلوغ‌ترین جاده‌ای که از آن گذشته عوارض می‌گیرد. اگر یک خودرو از جاده‌های $r_1, r_2, r_3, \dots r_k$ گذشـته باشد، از آن به میزان $\max(t^2_{r_1}, t^2_{r_2}, \dots, t^2_{r_k})$ عـوارض می‌گیرد که t_{r_i} تعـداد خودروهایی است که از جاده r_i در طول روز گذشته‌اند

مسیرهایی که رهیاب برای خودروهای مختلف (از متل قو تا تهران) پیشنهاد می‌کند لزوما یکسان نیست. همهٔ راننده‌ها از مسیری که رهیاب به آن‌ها پیشنهاد می‌دهند استفاده می‌کنند. اما اگر پس از پایان روز راننده‌ای بفهمد که می‌توانست از مسیر دیگری برود و عوارض کمتری بدهد، دیگر از رهیاب استفاده نخواهد کرد (چیزی که باب اسفنجی اصلا نمی‌خواهد اتفاق بیفتد) بنابراین، مسیری که برای هر راننده پیشنهاد می‌شود باید یکی از بهترین مسیرها باشد.

باب اسفنجی برنامهٔ دیگری طراحی کرده که در شروع روز، مقدار C، تعداد دقیق سفرها از متل قو به تهران در آن روز را پیش بینی می‌کند. حال، او از شما می‌خواهد که برنامهٔ رهیاب را به گونه ای ارتقا دهید که با گرفتن نقشهٔ جاده‌ها و عدد C، مسیرهایی را به هر راننده پیشنهاد دهد که با عوض کردن راه کسی نتواند پول کمتری بپردازد. می‌دانیم که تمام مسافران از متل قو تا تهران از برنامه رهیاب استفاده می‌کنند

ورودی

در ورودی چند آزمایه وجود دارد. هر آزمایه با یک خط شروع می‌شود که شامل پنج عدد طبیعی N، E، M، T و C است. اعداد صحیح N و E به‌ترتیب تعداد شهرها و جاده‌ها هستند ($0 \le E \le 1000$) ($2 \le N \le 500$). شهرها از ۱ تا N شماره‌گذاری شده‌اند. اعداد M و T که در ادامه می‌آیند، به‌ترتیب شماره شهرهای متل قو و تهران هستند ($1 \le M, T \le N, M \ne T$) عدد صحیح C تعداد مسافرت‌های بین متل قو و تهران در جمعه هست ($0 \le C \le 10^6$) در E خط بعدی، هر خط شامل دو عدد xi و yi است که نشان دهنده یک مسیر یکطرفه از xi به yi است. تضمین شده که حداقل یک مسیر از متل قو تا تهران وجود دارد. آخرین خط ورودی پنج عدد صفر داده می‌شود که پایان آزمایه هاست.

خروجی

برای هر آزمایه خروجی یک عدد است که جمع مبلغ عوارض هاییست که پرداخت می‌شود اگر تمام راننده‌ها از مسیرهای پیشنهادی پیروی کنند و برنامه شروط گفته شده را داشته باشد. می‌دانیم که این مقدار یکتا است.

مسئله ۱۱- ایستگاه‌های پایه

شرکت مخابراتی Viracell بزرگ‌ترین شبکه موبایل در Nevercity است. برای خدمت‌دهی بهتر به مشتریان، اپراتور تعداد زیادی ایستگاه فرستنده- گیرندهٔ پایه (BTS) در سطح شهر نصب کرده است. هر BTS در فرکانس ثابت مشخص شده کار می‌کند.

به دلیل گسترش سریع شبکه در سال‌های اخیر، کیفیت سرویس کاهش یافته و باعث اعتراضات زیاد از طرف مشتریان شده است.

در نتیجه ادارهٔ تنظیم مقررات تصمیم به تحقیق درباره این مشکل و آماده سازی یک گزارش گرفته است. تا این لحظه ادارهٔ تنظیم مقررات از اپراتور خواسته است تا ۲ گیرنده مخصوص را روی دو BTS نصب کنند تا داده‌های تمام BTSها را برای تحقیق بیشتر جمع‌آوری کنند. اپراتور آزاد است تا انتخاب کند که گیرنده‌ها را روی کدام BTSها نصب کند. به دلیل کمبود وقت اپراتور شما را استخدام کرده است تا به آن‌ها برای نصب گیرنده‌ها مشاوره دهید تا بهترین نتیجه کسب شود.

طبق یافته‌های شما، بهترین نتیجه وقتی کسب می‌شود که دو گیرنده کمترین اختلال سیگنال را داشته باشند. همچنین شما دو قانون ساده برای کم کردن اختلال سیگنال پیدا کرده‌اید. اول این که گیرنده‌ها باید روی دو BTS با فرکانس مختلف نصب شوند. دوم این که گیرنده‌ها باید تا جایی که ممکن است از هم دور باشند. به همین دلیل پیشنهاد شما به اپراتور این است که گیرنده را روی BTS هایی با بیشترین فاصله که فرکانس متفاوتی دارند نصب کنند. اپراتور از شما خواسته که این جفت را پیدا کنید.

ورودی

در ورودی چند آزمایه وجود دارد. خط اول هر آزمایه شامل عدد صحیح و مثبت n که تعداد BTS ها را نشان می‌دهد است.

($2 \le n \le 100,000$) در هر n خط بعدی، سه عدد مثبت x، y و k می‌آیند که (x, y) نشان دهنده مکان BTS و k نشان دهندهٔ فرکانس آن است. ($0 \le y, x \le 1000, 0 \le k \le 100$) هر شماره فرکانس به یکی از فرکانس‌های از پیش تعریف شده در شبکه اشاره می‌کند. فرض می‌شود که حداقل دو BTS با فرکانس‌های مختلف در ورودی وجود دارند. توجه کنید که BTSهای مختلف می‌توانند مکان یکسانی داشته باشند. ورودی با یک خط شامل صفر که نباید پردازش شود پایان می‌پذیرد.

خروجی

برای هر آزمایه، یک خط شامل مربع فاصله دورترین جفت BTS با فرکانس متفاوت، چاپ کنید.

مسئله ۱۲- اسکلت

شما به تازگی قبیله‌ای در بازی clash of clans تاسیس کرده‌اید. هر قبیله شامل تعدادی دهکده و تعدادی جاده یک طرفه است که دو دهکده را به هم متصل می‌کنند. فرض کنید تمام جاده‌ها هم‌طول هستند. شما می‌توانید در قبیله خود نیرو تعلیم دهید و سپس به قبیله‌ای دیگر حمله کنید. نیروهای مورد علاقه شما دیوار شکن‌ها هستند. (اسکلت‌هایی که بمب حمل می‌کنند، به سمت دیوار دهکده دشمن می‌روند و آن را می‌شکنند)

در حین یک حمله شما قبیله دشمن را انتخاب می‌کنید، به هر

دهکده از قبیله دشـــمن یک اســـکلت می‌فرستید و یک عدد حقیقی مثبت t برای زمان‌سنج بمب‌ها انتخاب می‌کنید و سپس دکمه حمله را می‌زنید. بلافاصله هر اسکلت به صورت تصادفی یک دهکده دشمن دلخواه را به‌عنوان هدف انتخاب می‌کند به‌طوری که هر دهکده، هدف دقیقا یک اســـکلت می‌شود. توجه داشته باشـــید که دهکدهٔ شروع و دهکدهٔ هدف اسکلت می‌توانند یکی باشند.

تمام اســـکلت‌ها در حین حمله با ســـرعت ثابت و مساوی یکدیگر و بدون توقف می‌دوند. آن‌ها جاده‌ها را در مسیر درستشان طی می‌کنند (جاده‌ها یک طرفه هستند) و می‌توانند از یک جاده چندبار بگذرند.

بعـــد از دقیقا t ثانیه انفجاری بزرگ رخ می‌دهد و تمامی اســـکلت‌ها منفجر می‌شـــوند. یک حمله موفقیت آمیز اســـت اگر تمام اسکت‌ها بتوانند اهداف خود را منفجر کنند. اســـکت‌ها بسیار باهوش هستند و هر کدام از آن‌ها مسیری را انتخاب می‌کنند که تضمین کند بعد از t ثانیه به هدف رسیده‌اند، البته اگر چنین مسیری وجود داشته باشد.

به شما لیستی از قبایل داده می‌شود، وظیفه شما پیدا کردن قبایلی از میان آن‌هاست که برایشان می‌توانید با زمان‌سنجی مناسب موفقیت حمله را تضمین کنید.

ورودی

در ورودی مشخصات چندین قبیله موجود است. برای هر قبیله، خط اول شـــامل تعداد دهکده‌هـــای آن ($1 \le n \le 50000$) و تعداد جاده ها ($1 \le m \le 10000$) اســـت. دهکده‌ها از ۱ تا n شماره‌گذاری شده‌اند. هر کدام از m خط بعدی ورودی از ۲ عدد صحیح x و y که با یک فاصله از هم قرار دارند تشکیل شـــده است که نمایانگر وجود جاده ای یک طرفه از دهکده x به دهکده y است. بین دو دهکده ممکن است بیش از یک جاده وجود داشته باشـــد و یا بعضی جاده‌ها می‌توانند دهکده مبدأ و مقصد یکســـانی داشته باشند. ورودی با خطی به شکل ۰ ۰ به پایان می‌رسد.

خروجی

برای هر قبیله در صورتی که می‌توانید با انتخاب t مناســـب موفقیت حمله را تضمین کنید نویســـهٔ Y را در خروجی چاپ کنید و در غیر این صورت نویسهٔ N را بنویسید.

زنگ تفریح

تعریف کلاس متخصص نرم‌افزار

```
Type
SoftwareProfessional = class
{
 double        salary;
 long          lunches;
 float         jobs;
 char          unstable;
 void          work;
 volatile      short_temper;
 union         none;
 enum          workdays(mon, tues, wed, thurs, fri, sat, sun)
};
EnhancedSoftwareProfessional= class (SoftwareProfessional)
{
    int       rovert;
    long      hair;
    void      other_humans;
};
```

زبان C

زبان C آنقدرها هم که بعضی‌ها می‌گویند سخت نیست. مثلاً دستور

void(*(*f[ ]) ( )) ( )

تابع f را به‌عنوان آرایه‌ای به طول نامشخص از اشاره‌گرهایی به توابعی که اشاره‌گرهایی به توابعی که مقدار تهی را برمی‌گردانند را بر می‌گردانند تعریف می‌کند.

تبلیغات در پیام‌های خطا

شرکت مایکروسافت دیروز اعلام کرد که در پیام‌های خطایی که در ویندوز ظاهر می‌شود، فضای تبلیغات در اختیار علاقه‌مندان قرار می‌دهد. بنابر آمارهای داخلی مایکروسافت، هر کاربر سیستم عامل ویندوز به طور میانگین روزی چند بار با پیام خطای ویندوز مواجه می‌شود و این فرصت خوبی برای نمایش آگهی‌های تبلیغاتی است.

آمارهای مایکروسافت نشان می‌دهد که در هر لحظه در جهان چند میلیون نفر پیام General Protection Fault را مشاهده می‌کنند.

مایکروسافت همچنین اعلام کرد که به زودی در صفحهٔ تمام آبی

که پس از فروپاشی (crash) ویندوز ظاهر می‌شود، بنر تبلیغاتی قبول خواهد کرد.
وزارت دادگستری آمریکا بلافاصله اعلام کرد که بررسی می‌کند آیامایکروسافت به خاطر کنترل انحصاری که بر روی پیام‌های خطایش دارد، با این اقدام، قانون ضدانحصار را نقض خواهد کرد یا نه.

چه کسی برای جرّاحی بهتر است؟

۴ جراح با هم صحبت می‌کردند.
اولی گفت: من ترجیح می‌دهم حسابدارها را عمل کنم. چون وقتی شکمشان را باز می‌کنی، همه چیز شماره‌بندی شده است.
دومی گفت: من کتابدارها را ترجیح می‌دهم. همه چیز درون شکمشان به ترتیب حروف الفباست.
سومی گفت: معلوم می‌شود تا حالا برقکارها را عمل نکرده‌اید! همه چیز درون شکمشان با رنگ‌های مختلف به خوبی قابل تشخیص است.
چهارمی که تا این لحظه ساکت بود و به حرف‌های همکارانش گوش می‌داد گفت: من مهندسان را بر همهٔ آن‌ها که گفتید ترجیح می‌دهم زیرا وقتی عمل تمام شد و شکمشان را بستید، اگر چند تا از اعضای بدنشان زیاد آمد و بیرون ماند، این مسئله را به خوبی درک می‌کنند.

وراثت چندگانه

سوال: چرا برای ثروتمند شدن، C++ بهتر از جاوا و C# است؟
جواب: چون از وراثت چندگانه پشتیبانی می‌کند.

مصاحبهٔ استخدامی

شرکت ما برای استخدام برنامه‌نویس آگهی داده بود.
از یکی از کسانی که برای مصاحبه دعوت شده بود پرسیدم: شما می‌توانید برنامه‌ای به زبان C بنویسید که یک عدد تصادفی را چاپ کند؟
جواب داد: البته که می‌توانم. بگوئید چه عددی مورد نظرتان است تا برنامه‌اش را بنویسم.

نکتهٔ آخر

مغز آدم به‌طور معمول با ده درصد ظرفیتش کار می‌کند. بقیه‌اش سربار سیستم عامل است.

لیست اعضای حقوقی

انجمن انفورماتیک ایران

اعضای حقوقی فعال در حوزه راه‌حل‌های برنامه‌ریزی منابع سازمان و نرم‌افزارهای پیشرفته سازمانی

- آریا همراه سامانه
- آفرینش رایانه کیهان
- ایریسا (بین‌المللی مهندسی سیستم‌ها و اتوماسیون)
- بهکو (بهینه کوشان سپهر)
- برگ سیستم پویا
- پارس رویال نفیس
- پارس اوک کیش
- پارس تصمیم
- پردازش موازی سامان
- پردازش اطلاعات ریسمان
- توسعه یکپارچه ایلیا
- تدوین فرآیند
- تحقیق و توسعه ارتباط
- چارگون
- خدمات مهندسی عصر اندیشه
- درگاه ارتباطات جدید
- دی رایانه نوین
- دانش افزار نارون شریف
- راهکارهای برتر یکپارچه ساپارد اروین
- رایاوران توسعه
- مهندسی رای دانا آفرین
- سامانه پرداز اریس
- پایه‌ریزان راه‌کارهای فراگیر
- سامه آرا پردازشگر
- سبز داده افزار
- سیستم‌های اطلاعات مدیریت شرق رایا
- سند پرداز
- شماران سیستم
- گروه نرم‌افزاری پیوست
- گلرنگ سیستم
- فناوری نوین رایا شریف
- فناوری اطلاعات و ارتباطات پاسارگاد آریان (فناپ)
- فاوا نفت صبا کیش
- همکاران سیستم مدیریت طرحهای عمومی
- نوید ایرانیان (گسترش فضای مجازی)
- مانیر (مشاورین انفورماتیک نیرو)
- مجتمع داده‌ها و سیستم‌ها (MDS)
- مهندسی بهینه ایران
- ماموت مدیریت سیستم‌ها
- مهندسی نرم‌افزاری رایورز
- مهندسی نرم افزار فرارای (سهامی خاص)
- نماد ایران
- نسیم ارتباط دماوند
- داده پردازی نیلرام مانا
- گروه مشاورین ورانگر نوین
- گروه توسعه فناوری اطلاعات بصیر

لیست اعضای حقوقی

انجمن انفورماتیک ایران

مشاوران یام آذر

نوماتک

اعضای حقوقی فعال در حوزه نرم‌افزارهای کاربردی

آریا ایمن تدبیر

اطلاع‌رسانی پیوند داده‌ها

ایران رایانه

امن پرداز ان سورنا

آریانا پرداز آینده (آرپا)

انتقال دانش صنعت انرژی برق

پندار کوشک ایمن

پژند الکترونیک

پردازش اطلاعات و ارتباطات هاماوران آسیا

پیک فرزانه پویان

پاریزان صنعت دانش

پیشگامان فن آوری هوداد

توسعه ارتباطات رایانه‌ای آبانگان

توسعه فناوری اطلاعات جهان افزارنوین

تدبیرگران نوآوری رایسان

توسعه زیر ساخت ماهان تک

تیم تعمیرات و نگهداری خطوط لوله خاورمیانه

تحلیلگران فن آوری پویش نور

چشم انداز تجارت به دان

داده پردازان احداث

داده‌پردازان آبشار

داده‌پردازی حرفه‌ای‌ها

دی کاو ماندگار

راهبر نیروی خراسان (رانیر)

طرفه نگار

رایانه خدمات امید

سافت سیستم کیش

سنجه حساب

سبزافزار آریا

شایگان سیستم

صنایع فن آوری طراحان بهینه

مهندسی فرا طرح نوین مهرگان پیشتاز

مهندسی تکرو سیستم

مدیران سیستم پاسارگاد

مهندسی نرم‌افزار ایده‌گستر پوریا

فاوا هزاره کیش

فراکنش

فناوری ارتباط امن خاورمیانه

همراهان سیستم گوهر

همکاران سیستم

نرم‌افزاری امن پرداز

اهورا سیستم اهواز

رهیاب ریان فردا

کهکشان دانا

مدار گسترش فناوری اطلاعات

لیست اعضای حقوقی

انجمن انفورماتیک ایران

موسسه فرهنگی دیجیتال تدارک نرم‌افزار پایتخت پارسیان

کارانس ایرانیان

گروه شرکت های مهندسی نرم افزار فراپیام

گام الکترونیک

حامی سیستم شریف

اعضای حقوقی فعال در حوزه بانکی و بانکداری الکترونیکی

الماس هوشمند ایرانیان

آسمان صبح فردا

ارتباطات هدی ارقام

بهسازان ملت

بانک اقتصاد نوین

بانک آینده

به‌پرداخت ملت

پدیسار انفورماتیک

پردازشگران سامان

پرداخت اکترونیک سداد

پویا

بانک سرمایه

توسعه‌فن‌افزارتوسن

توسعه فناوری رفاه پردیس

توسعه فناوری تجارت حکمت

توسعه خدمات الکترونیکی آدونیس

ثامن ارتباط عصر

شرکت کسب و کارهای نوپای خاورزمین

ماتریس تحلیلگران سیستم‌های پیچیده

خدمات انفورماتیک

توسعه سامانه‌های نرم‌افزاری نگین (توسن)

خدمات ماشین‌های اداری امیم رایانه

سامان امنیت پرداز کیش

سفیر آبی آرام

سامانه یکپارچه سیمرغ تجارت

سامانه و ازریزی

شرکت رایان صنعت اقتصاد نوین

شرکت توسعه فن‌افزار توسن

داده‌پردازی ایران

داده پردازان ماهان شبکه جنوب

داده‌ورزی فرادیس البرز

داتیس آرین قشم

شرکت شبکه الکترونیکی پرداخت کارت شاپرک

سامانه تبادل الکترونیک دفاتر پیشخوان دولت

فناوران تجارت الکترونیک الماس

مشاورین بهبود روش‌ها و سامانه‌های مبنا

فناوری اطلاعات ناواکو

فناوری اطلاعات ارتباطات پارسه بین الملل اتیه وندادد

مدیریت و مهندسی کرون‌راد

گرایش تازه کیش

گسترش فناوری‌های نوین کشاورز

مهندسی آتی ساز اب ایرانیان

توسعه فناوری اطلاعات خوارزمی

لیست اعضای حقوقی

انجمن انفورماتیک ایران

بانک سینا

هوشمند الکترونیک
الماس شرق کیش

اعضای حقوقی فعال در حوزه اینترنت و پورتال‌ها

آریا رسانه نوین تنکابن

آوای هوشمند زورق آرامش

ارتباطات فن‌آوا

ارتباطات مبین نت

افرانت

پارس لین

پیشگامان دامنه فناوری

توسعه آرمان ارتیاش تاریانا

تندیس تلاش و تفکر

توسعه فناوری ریز فن پردازش پرشین

تبادل الکترونیک فرداد

تدبیر (ایران سون استار)

فناوران اهل قلم نوین

دنیای ایده های تابان فناوری

داده پردازی پویای شریف

دانشگاه آزاد اسلامی واحد سوسنگرد

رایان هوشمند نویان

فناوری اطلاعات و ارتباطات پارس
پردازش سدید

فن‌آوری اطلاعات پارسیان میزبان

فناوری رادین پاسارگاد

کیانا پارسیان کیش

کلیدگستر آینده (نت بینا)

گروه شرکت‌های فن‌آوا

مؤسسه گسترش اطلاعات و ارتباطات
و فرهنگی ندا رایانه

هاست ایران

نویان ابر آوران

اعضای حقوقی فعال در حوزه شبکه و سخت‌افزار

آوید رایانه هیراد

آداک فناوری مانیا

آلبالو رایانه سخت افزار

آرین وب ایرانیان

ایده نگر نوین اطلس

پارس فایبرنت

پیشتازان اندیشه پویا

اروند رایان گستر

برد پرداز رایانه

پرشیا پرداز آسیا

پارسیان فیبر ارتباط

پرتو بیتا جاوید

پیشرو توسعه نوآوران آرتا

توانش

ترویج صنعت سومی پارسیان

تحلیلگران شبکه گستر پارسه

تحلیلگران اطلاعات نگاره

تحلیلگران ارتباط ایرانیان

توسعه سرمایه گذاری گروه آروند

لیست اعضای حقوقی

انجمن انفورماتیک ایران

توسعه داده‌پردازی بنیس

دانشگاه آزاد علوم تحقیقات خوزستان

دانش رایان ارتباطات روز

رایان مهر الکتریک

رهنمون فناوری اطلاعات

رهروان طب ثمین

خدمات کامپیوتری خانه سیستم

خدمات رایانه ای بهینه پرداز پویا

خدمات آواژنگ

چرخه ارتباط سبز

سرو رایانه

سامانه هوشمند سورنا

سازگار الماس شرق

سیمرغ سامانه تهران

فاوا موج

فناوری اطلاعات سهلان

فرا اتصال رایان عصر

فرصت کیش

فدک رایان

مهندسی پیمان عمران نیرو

مهندسی رادین راهبرد رایانه

موسسه آموزشی نکوئی

شایان توسعه البرز

شیوه نوین ارتباط پایدار

شبکه پایدار فناوری اطلاعات

شبکه هوشمند پدیده آپادانا

صبا کاربردلتا

شرکت مهندسی گسترش ارتباطات نو خاورمیانه

شبکه گستر نسل جدید

فن‌آوری اطلاعات و ارتباطات آرمان تندیس ایرانیان (فن آرا)

فناوران آدرین داده پرداز

کارنما رایانه

کیسان صنعت ایرانیان

کیسون

گروه مهندسین فرایند

گروه بازرگانی آریانا پارس راژمان

گروه دا ده ورز جویا

مهندسی رایان توان افزار

مهندسی شبکه گستر

مهندسی پارتیان ابتکار پایدار

مهندسی افق داده ایرانیان

نوین ارتباط نوآوا

شرکت توانش

اندیشه‌پرداز دوران

گروه تجارت اوژن کیش

انتقال داده پرشین

ویژن پلاس اروپا

ویرا صنعت پارسه

نوآوران اندیش رایانه غرب

همجوا رایانه تهران

لیست اعضای حقوقی

انجمن انفورماتیک ایران

اعضای حقوقی فعال در حوزه مدیریت پروژه

- آتی سپهر توسعه آریانوآور
- فناوران اطلاعات بهاران
- اندیشه وران
- سامان اندیشه ستیا
- نفت ایرانول

اعضای حقوقی مشتریان بهره‌بردار از راه‌حل‌های نرم‌افزارهای پیشرفته سازمانی و بهره‌بردار از فناوری اطلاعات

- آوای فناوری اطلاعات سلامت
- ارتباطات نت میهن
- اتصال صنعت میانه
- تعاونی آموزشی خدماتی رایانه ای پیام دانش شهرکرد
- بهسازان صنعت و اطلاعات
- پارسین تجارت پایا کیش
- پتروسان کالا نماد
- توزیع برق آذربایجان
- توسعه راهکارهای نور مبین (ترنم)

- پردازش هوشمند البرز
- خدمات مشاور خرد پیروز
- زورق آرامش
- مهندسی و ساخت بویلر مپنا
- گروه کارخانجات چینی مقصود
- گسترش فناوری اطلاعات ستاره ویدا
- مهندسی آریا تدبیر ایلیا
- طبیعت‌زنده

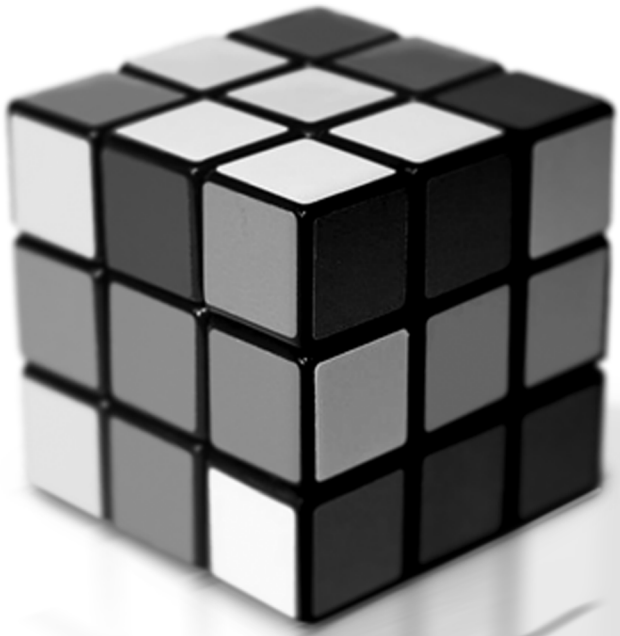
اعضای حقوقی فعال در حوزهٔ آموزش و پژوهش

- آماج فناوری اطلاعات وارتباطات ایرانیان
- آزمایشگاه آزمون وتایید نرم افزار
- آموزشکده دخترانه سما- واحد کرج
- آموزشکده فنی و حرفه ای سما واحد گرگان
- الماس رایان ایرانیان
- سماتک

- فناوری اطلاعات بین‌الملل
- مؤسسه عالی مدیریت دانش
- کهکشان نور
- شرکت علم یاران
- پرداز گستر تدبیر
- طلیعه کاوش
- گروه فنی مهندسی ایده سازان مبتکر قرن

اعضای حقوقی فعال در حوزهٔ مشاوره

- آرین
- الهه کوچک نرم افزار
- تیوا سیستم
- مشاوره مدیریت رایزن سامانه گستر
- موسسه حقوقی واقتصادی آرمان ایرانیان
- طرح و توسعه الکترونیک افق
- سرآمد فناوری اطلاعات
- مشاوره مدیریت پیشداد سرویس
- مهندسی کاربرد سیستم سدید



قابل توجه مدیران عامل، مدیران فروش
مدیران بازرگانی و مدیران تبلیغات شرکت‌ها

به اطلاع می‌رساند که در زمینه‌های زیر آمادگی همکاری با آن موسسه محترم را داریم:

طراحی و چاپ انواع فرم‌های چاپی (بولتن، بروشور، کاتالوگ، فولدر، سربرگ، پاکت و...)
طراحی و تهیه انواع پلات اعم از بنر، کتدلمینت، فلکسی‌پیس، رول آپ، پاپ آپ و...
طراحی، چاپ، صحافی سررسید، دفترچه تلفن، دفترچه یادداشت به صورت اختصاصی و...
تهیه و تولید انواع ساک‌های تبلیغاتی دستی ویژه نمایشگاه‌ها (برزنتی، نایلونی، سوزنی و...)
تهیه و ارائه انواع هدایای تبلیغاتی اعم از کیف، کلاسور، خودکار، ساعت، البسه و... به صورت اختصاصی

لازم به توضیح است که بنا به درخواست شرکت در هر یک از موارد فوق
طبق تعرفه‌های روز قیمت اجناس و خدمات محاسبه و اعلام خواهد شد
و در صورت توافق برگه سفارش تنظیم و کار ارائه خواهد گردید.

email:kargahhonar@yahoo.com
تلفن: ۸۸۳۲۸۴۱۷-۲۱