به نام خدا

تمرین سری اول
مهندسی نرم‌افزار۲

نعیم اصفهانی
۸۴۲۰۱۰۰۳

سئوال ۱:

۱. الگوریتم
نظریات مربوط به پیچیدگی و محدودیت الگوریتم‌ها. در صورت نیاز به داشتن یک الگوریتم کارا می‌توان تعیین کرد که آیا الگوریتم مورد نظر کارایی دلخواه را دارد یا نه. در ضمن اگر بتوان مساله را به رده‌ی مسائل NP-C کاهش داد، می‌توان مطمئن بود که راه حل سر راستی برای آن تا کنون پیدا نشده و از تخصیص وقت بیشتر جلوگیری کرد.

۲. مدیریت پروژه[×]
نظریات و مدل‌های ارائه شده در این علم می‌توانند راه‌گشای مدیر پروژه در برنامه‌ریزی، تخصیص بودجه، زمان‌بندی، شکستن کار و ... باشد. تکنیک‌های زمان بندی، پیش‌بینی و ... همه نظریاتی علمی در این قالب هستند.

۳. ریاضیات گسسته و روش‌های صوری
در سیستم‌های بسیار بزرگ که با جان مردم یا سرمایه‌ی بسیار زیاد سر و کار دارند، باید از کارکرد سیستم مطمئن بود. این کار با استفاده از نظریات و راه‌کارهای موجود در روش‌های صوری و ریاضیات گسسته انجام می‌شود.

۴. مدل‌سازی
در برخی از سیستم‌ها که نمی‌توان محیط واقعی را به‌طور کامل در اختیار داشت و یا محیط واقعی بر اثر بزرگ بودن قابل بررسی نیست باید از مدل‌سازی محیط استفاده کرد که در این راستا از نظریات فراوان این شاخه‌ی علمی استفاده می‌شود.

۵. تحلیل سیستم‌ها
برای تحلیل سیستم موجود از نظریات موجود در این شاخه‌ی علمی استفاده می‌شود. همچنین از انبوه تجربیات موجود در این شاخه‌ی علمی می‌توان استفاده کرد.

۶. روان‌شناسی
از نظریه‌های علمی فراوانی برای مجاب کردن خریدار و یا کارفرما استفاده می‌شود که از آن جمله می‌توان به نظریه‌ی بازی‌ها اشاره کرد که سعی می‌کند روابط انسان‌ها را در قالب بازی‌های بیان کند. این علم بیشتر در ارتباط با انسان‌ها موثر است؛ این انسان‌ها هم می‌توانند از کارکنان باشند و هم از مشتریان.

۷. هنر

تئوری رنگ‌ها برای طراحی واسط کاربری می‌تواند مناسب باشد. این‌که طراحی به‌گونه‌ای باشد که اولا تاثیر مطلوبی داشته باشد و در ثانی افراد مشکل‌دار مانند افراد کوررنگ را در نظر گرفت.

۸. اقتصاد

قانون پرتو "۸۰٪ کارایی توسط ۲۰٪ سیستم تامین می‌شود"

این قانون در برنامه‌ریزی جهت انجام کارها و اختصاص سرمایه استفاده می‌شود. مورد استفاده‌ی دیگر این قانون در زمانی است که نیاز به کاهش هزینه‌ها وجود دارد.

× در مواردی که نظریات علمی موجود فراوان هستند(حداقل از نظر نگارنده) توضیح در مورد نظریه‌ی علمی خاصی داده نشده‌است.

سئوال ۲:

۱. تعداد کل فایل‌ها در یک پروژه
۲. تعداد کل کلاس‌های سطح بالا در یک کلاس
۳. تعداد کل خطوط پیش‌پردازشی(مانند #include)
۴. تعداد کل خطوط کامنت
۵. تعداد خطوط سورس برنامه (بدون فاصله و کامنت)
۶. تعداد متدها در تعریف یک کلاس
۷. تعداد کلاس‌های دیگری که متدهای آن‌ها و یا متغیرهایشان توسط متدهای کلاس استفاده می‌شود
۸. تعداد gotoهای برنامه
۹. تعداد کارهای انجام شده
۱۰. تعداد درخواست‌های تغییر
۱۱. تعداد اصلاحات
۱۲. تعداد نیازمندی‌ها
۱۳. تعداد خطاهایی که در نرم‌افزار منتشر شده وجود دارد
۱۴. مدت زمان تولید نرم‌افزار
۱۵. اختلاف هزینه‌ی پیش‌بینی شده و واقعی
۱۶. میزان مستندات
۱۷. تعداد ورودی‌های برنامه
۱۸. تعداد خطوطی که توسط آزمون پوشیده شده‌اند
۱۹. تعداد خطاهای یافته شده در آزمون نسبت به فاز قبلی
۲۰. درصد تست‌های موفق

سئوال ۳:

ریسک‌هایی که در نظر گرفته نشد و در آخر پروژه با آن‌ها برخورد شد:

۱. عدم تعیین دقیق حدود پروژه: وظیفه‌ی پیاده‌سازی طراحی یک شرکت با ما بود، در آخر پروژه مشخص شد طراحی آن شرکت مورد تایید کارفرمایش نبوده و ما ناچار مجبور به تغییراتی شدیم.

۲. ترک نیروی انسانی: کل پروژه وابسته به طراحی بود که کل طراحی را انجام می‌داد ولی بسیاری از موضوعات را مکتوب نکرده بود. او در میان پروژه به دلیل مشکلات شرکت را ترک کرد و هزینه‌ی زیادی بر گردن شرکت افتاد.

۳. عدم بررسی دقیق ابزارها: اول بنابر این بود که از محیط برنامه‌نویسی مجتمع Eclipse به همراه یکی از امکانات کنار آن به نام MyEclipseIDE استفاده شود ولی این امکان مقیاس پذیر نبود و با بزرگ شدن پروژه مشکلاتی را فراهم کرد.

ریسک‌هایی که در نظر گرفته شدند:

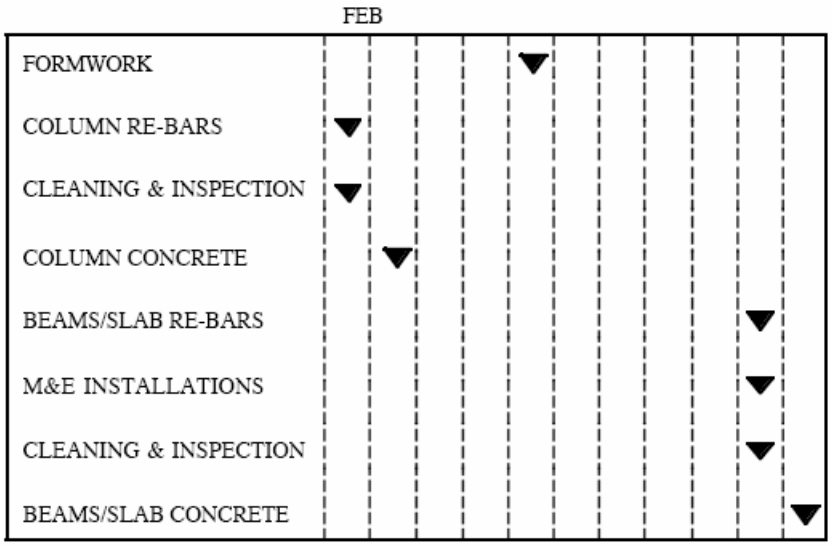
۱. عدم در دسترس بودن کارمندان: به دلیل دانشجو بودن کارکنان پیش بینی می‌شد که از ابتدای سال تحصیلی آن‌ها به طور کامل در دسترس نباشند و بنابراین برنامه‌ریزی شد و به دلیل پیش‌بینی شدن مشکل حادی برای پروژه پیش نیامد.

۲. عدم رضایت کاربر از ظاهر سیستم: پیش بینی می‌شد که کاربر از ظاهر سیستم نهایی در نهایت ناراضی باشد و ظاهر به گونه‌ای نوشته شد که منعطف باشد و قابل تنظیم توسط خود او باشد و این موضوع بسیار کمک کرد.

سئوال ۴:

نمودار GERT ، نمودار Milestone ، شبیه‌سازی Monte Carlo. استفاده از تکنیک WBS.

نمودار Milestone:



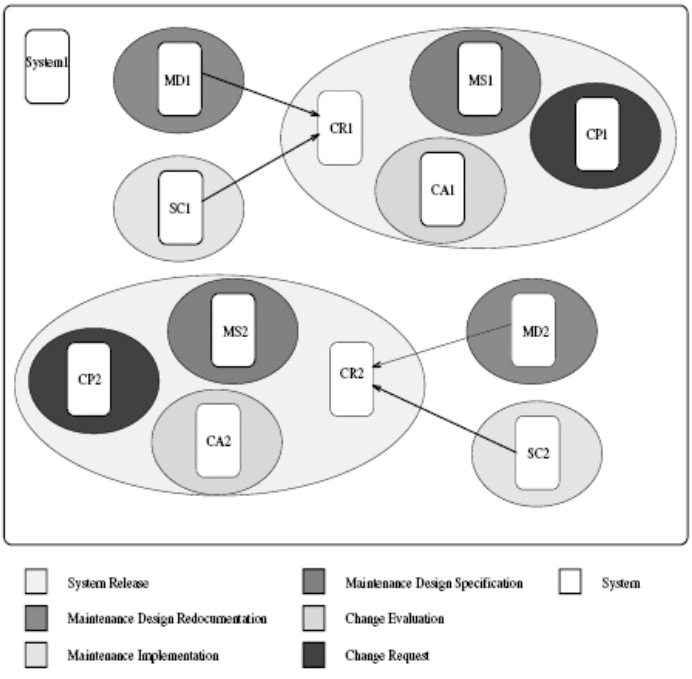
نمونه‌ای از نمودار Milestone

ساده‌ترین برنامه‌ریزی برای یک پروژه توسط نمودار Milestone فراهم می‌شود. این نمودار شامل لیستی از کارها، انجام دهنده‌ی آن‌ها و زمان پیش‌بینی شده برای اتمام کار است. زمان اتمام کار توسط برنامه‌ریز پروژه تعیین می‌شود. در این نمودار وابستگی کارها درنظر گرفته می‌شود ولی نمایش داده نمی‌شود. به دلیل سادگی، این نمودار خیلی استفاده می‌شود ولی کاربرد آن نسبتا محدود است و ابزار ضعیفی به حساب می‌آید.
این نمودار به مجری پروژه این امکان را می‌دهد که تخمینی از زمان پایان کار داشته باشد. در زمان اجرای هر کار تنها اطلاعات کیفی در مورد روند پیشرفت کار در دسترس است و تنها در پایان انجام هر کار است که می‌توان دقت برنامه‌ریزی را سنجید: یا کار مطابق برنامه‌ریزی پیش رفته است یا نه.

سئوال ۵:

مدل مورد نظر در این بحث مدلی است ادراکی به نام CONFORM[1] [1]. هدف اصلی این مدل این است که روند عملیات نگهداری و تغییرات حاصله را تحت کنترل در آورده، مستندسازی را به صورت افزایشی انجام دهد. با استفاده از این مدل جلوی تغییرات به صورت تصادفی را گرفته و اجازه‌ی دسته‌بندی و تصمیم گیری در مورد تغییرات را می‌دهد. مدل ادراکی در این مدل بر پایه‌ی مدل داده‌ای ORM[2] است. این مدل ادراکی از مقاله‌ای (این مقاله در CD ضمیمه شده است) که در سال ۱۹۹۷ ارائه شد استخراج گشته است. مدل داده‌ای مذکور شباهت زیادی به مدل شیء گرا دارد و یکی از تفاوت‌های آن با مدل شیء گرا در این است که روابط شهروند درجه اول هستند.

این مدل ادراکی سعی می‌کند ساختاری به روند تغییرات بدهد و پایه‌ی آن فرم‌های درخواست تغییر هستند که به صورت یک شیء در می‌آیند و این شیء با توجه به تغییراتی که می‌کند اشیاء مختلف و مرتبطی می‌سازد. ارتباطات در این مدل به گونه‌ای هستند که اشیاء مرتبط در یک فاز تغییر را قابل ردیابی کنند. توضیح مفصل این مدل در مقاله‌ی ضمیمه آمده‌است و در ادامه تنها به بیان مثالی که در قالب همان مقاله آمده اکتفا می‌کنیم.



نمایش شکلی مثال انتشارهای مختلف یک سیستم

[1] Configuration Management Formalization for Maintenance
[2] Object Representation Model

در شکل دسته‌های اشیاء (بیضی‌ها) و فرم‌ها (مستطیل‌ها) دیده می‌شوند. سیستم ۱ نام سیستم نرم‌افزاری‌ای است که توسط مدل ما نگهداری می‌شود و چندین انتشار[3] دارد. اولین انتشار ChangeRequest1 را تولید کرده‌است که قسمت‌هایی از نرم‌افزار را که تغییر کرده‌اند به همراه تاریخچه‌ی تغییرات را در بر می‌گیرد. تغییرات این انتشار در ابتدا توسط ChangeProposal1 پیشنهاد شده‌اند و پس از تایید ChangeApproval1 حاصل شده است. MaintenanceSpecification1 هم برای نگهداری تاریخچه‌ی تغییر حاصل شده است. این تغییرات باعث طراحی‌ها و کدهایی شده است که در ModuleDesign1 و ModuleSourceCode1 منعکس می‌شود. به این ترتیب در هر انتشار یک نمونه طراحی و کد داریم و برای اطلاعات بیشتر می‌توانیم به دسته‌ی اشیاء مرتبط مراجعه کنیم.

[1] Miriam. A. M. Capretz. "Conceptual Model for a Software Maintenance Environment," *hicss*, p. 64, 30th Hawaii International Conference on System Sciences HICSS Volume 5: Advanced Technology Track, 1997. DOI= http://doi.ieeecomputersociety.org/10.1109/HICSS.1997.663160

[3] Release