مهندسی نرم‌افزار

تمرین اول

دانشکدهٔ مهندسی کامپیوتر

دانشگاه صنعتی شریف

استاد:

جناب دکتر مهران ریواده

گروه ۵

امیرمحمد قاسمی

سید علیرضا هاشمی

حمیدرضا کامکاری

یگانه قره‌داغی

نیما سالم

سؤال ۱

سناریو ۱:

"یک سیستم جدید برای حفاظت از کشور" (۰)
"تا کنون چنین سیستمی ساخته نشده و منابع اطلاعاتی برای این سیستم وجود ندارد" (۱)
"سیستم بزرگ و پیچیده است و تولید آن میتواند به ده سال زمان احتیاج داشته باشد" (۲)
"پژوهشگران ایده مبهمی برای ساخت آن دارند" (۳)
"تعداد ذینفعان و محدودیت‌هایی که بر این نوآوری تاثیر گذار خواهند بود نیز زیاد است" (۴)

نوآوری که قرار است انجام شود، بسیار بزرگ است و مستلزم فرآیندهایی است که توانایی مدیریت پروژه‌های بزرگ را داشته باشند.
از آن جایی که طبق (۴) ذینفع‌های متعددی وجود دارد و محدودیت‌های بسیاری بر پروژه حاکم است، ارتباط فراوان و مکرر با ذینفع‌ها به شدت لازم میباشد. همچنین مبهم بودن نحوه ساخت (۳)، نیز نیاز به دریافت بازخورد و تعامل را با ذینفع‌های پروژه و پژوهشگران مضاعف میکند.

توسعه چنین فناوری گسترده‌ای در مدتی طولانی استفاده از یک فرآیند تکاملی را بدیهی میکند. یعنی فرآیند باید iterative و incremental باشد که از ویژگی‌های فرآیندهای تکاملی هستند. Iterative بودن باعث میشود که خروجی هر iteration قابل نمایش باشد و بنابراین میتواند باعث رفع ابهام بین ذینفع‌ها شود. همچنین دید پژوهشگران با پیشرفت نرم‌افزار تکامل میابد و اصلاح میشود.

با توجه به (۱) و (۳)، اعمال تغییرات در پروژه بسیار بدیهی مینماید. چرا که چنین سیستم تا به حال ساخته نشده است. پس با پیشرفت فرآیند، کم‌کم جزئیات و نیازمندی‌های پروژه معلوم میشوند و دید پژوهشگران از پروژه شفاف تر میشود و چالش‌ها خودشان را نشان میدهند.

با توضیحات داده شده، دو کاندید فرآیند حلزونی و چابک بر روی میز قرار دارند.

با توجه به امنیتی بودن پروژه و mission critical بودن آن طبق (۰) و موجود بودن ریسک زیاد در انجام پروژه به خاطر نوآوری آن و هزینه‌ی بسیار بالای ریسک در مقایس‌های نظامی و امنیتی، فرآیند حلزونی گزینه‌ی مناسب‌تری برای انجام این پروژه میباشد. همچنین مستندسازی رسمی و تاکید بر روی فرآیند یکی دیگر از نیازمندی‌هاست که فرآیندهای چابک فاقد آن میباشند.

سناریو ۲:

"این شرکت تجربه تولید این سیستم نرم‌افزاری را در مقایس بزرگ نداشته است" (۰)
"بیمارستان از تفاوت‌های ریزی که بین کشورهای مختلف وجود دارد اطمینان و آگاهی ندارد" (۱)
"تعداد کمی از مکان‌ها از این نرم‌افزار بهره‌مند خواهد شد" (۲)
"خواستار سازگاری در سیستم است" (۳)

در این سناریو از فرآیندهای چابک برای تولید سامانه استفاده میکنیم چرا که طبق (۰)، شرکت چارچوب این نوع سامانه‌ها را میشناسد و کافی‌است تا فرآیندش را برای تولید این سامانه‌ها در مقیاس بزرگ بهبود دهد. ویژگی که در فرآیندهای چابک مانند اسکرام پیدا میشود. سپس این نرم‌افزار تولید شده در مکان‌های محدود مورد استفاده قرار میگیرد و نکات ریز خود به خود پیدا میشوند و ایرادات و بازخورد دوباره به چرخه تولید نرم‌افزار تزریق میشوند تا نرم‌افزار تکامل پیدا کند.

سناریو ۳:

نکته‌ی مهم قابل برداشت در سناریو داده شده، این است که شرکت تولید کننده نرم‌افزار "چندین بار این سیستم نرم‌افزاری را برای سازمان‌های مشابه تولید کرده و با آن آشنایی کامل دارد". پس عملا سیستم‌های مشابه موجودند و شرکت در تولید آن‌ها دستی در آتش دارد.

در درجه نخست آشنایی و حتی تسلط کافی شرکت نرم‌افزاری بر چنین نرم‌افزارهایی و در درجه دوم وجود سیستم‌های مشابه باعث میشود که تمام stake holder ها اعم از سازمان و شرکت هر دو آگاهی کافی از نیازمندیهای محصول داشته باشند. بنابراین فرآیند پیشنهادی ما، فرآیند آبشاری (Waterfall) است.

سؤال ۲

تشابهات

هر دوی XP و Scrum فرآیند توسعه را به تعدادی sprint تقسیم میکنند. در هر دو قبل از شروع توسعه، جلسات برنامه‌ریزی دارند که user story ها در آن‌ها مشخص میشوند.
همچنین در هر دو جلسه برنامه‌ریزی قبل از شروع هر sprint داریم. هدف اولیه هر دو مشابه میباشد که عبارت است از تحویل نرم‌افزار با کیفیت در سریع‌ترین زمان ممکن به مشتری.

تفاوت‌ها

تفاوت‌های Scrum و XP را میتوان در تعدادی حوزه برجسته کرد

۱. هدف اصلی

تفاوت اصلی بین Scrum و XP در این مورد میباشد. Scrum شدیدا متمرکز بر خود عمل مدیریت کردن است و فعالیت‌های جانبی (به جز کد زدن) در آن بسیار فراوان میباشند و در عوض تاکید کمتری بر روی جنبه فنی و مهندسی کار دارد یا این که اصلا چگونه محصول تولید میشود.

Scrum تعیین میکند که چگونه برنامه‌ریزی و تحلیل شود. Scrum بیشتر با کارایی تیم و قابل استفاده بودن محصول تولید شده در انتهای sprint درگیری است. همچنین Scrum نقش‌های مشخص، رخدادهای معلوم و مصنوعات متفاوت و گسترده‌ای دارد.

XP بر روی جنبه‌های مهندسی تمرکز بیشتری میکند به همین خاطر بهروش‌های چابک مانند pair programming و بهروش‌های تست نرم‌افزار در آن حضور پررنگی دارند. XP با فعالیت‌های که در بیانیه آن وجود دارد، نرم‌افزار با کیفیت فنی بالایی تولید میشود.

۲. Sprintها

یکی از مهمترین اصول روش‌های چابک، ارائه نرم‌افزار قابل تحویل در زمان‌های کوتاهی به نام sprint است. هر دوی فرآیندها از sprint به مراحل ایجاد نرم‌افزار استفاده میکنند. در Scrum مدت زمان sprintها ۲ الی ۴ هفته میباشد و طول آن‌ها منطعف است. در XP اما sprintهای کوتاه تر ۱ هفته‌ای (و بعضی مواقع ۲ هفته‌ای) وجود دارد. هدف XP انتشار محصول در یک sprint نیست بلکه هدف آن تولید یک نرم‌افزار بدون خطاست درحالیکه در scrum در انتهای یک sprint یک محصول قابل تحویل ارائه میشود.

۳. گنجاندن تغییرات

در Scrum، زمانی که تسک‌هایی که برای یک sprint مشخص شدند، دیگر امکان تغییر آن‌ها و اضافه یا کم کردن به آن‌ها وجود ندارد. XP انعطاف بیشتری در این مورد دارد. تغییرات در تسک‌ها توسط مشتری و هر زمانی میتواند انجام شود حتی در میان sprint اگرچه بهتر است که این تغییرات در مراحل ابتدایی و قبل از شروع sprint انجام شوند.

۴. Product owner

در Scrum، وظیفه ارتباط با product owner توسط scrum master انجام میشود که وظیفه دارد از واضح بودن تسک‌ها برای تیم توسعه اطمینان حاصل کند و به product owner در اولویت بندی product item backlog کمک کند.

در XP، مشتری مستقیما با تیم توسعه ارتباط برقرار میکند و user storyها را اولویت بندی میکند. همچنین مشتری است که به تیم توسعه بازخورد میدهد.

۵. اولویت‌بندی تسک‌ها

در Scrum، اولویت بندی تسک‌ها به عهده product owner است. ترتیب انجام تسک‌ها در یک sprint به عهده خود توسعه‌دهندگان است. آن‌ها میتوانند به هر ترتیبی که دوست دارند تسک‌ها را انجام دهند.

در XP، در مورد انجام تسک‌ها انعطافی وجود ندارد. تسک‌ها به همان ترتیبی که توسط مشتری مشخص شده‌اند باید انجام شوند.

سؤال ۳

در مدیریت و توسعه چابک، story point یک معیار انتزاعی برای تخمین دشواری هر user story است و پیچیدگی‌ها، خطرات و تلاش‌های لازم برای انجام آن را در بر میگیرد. استفاده از story point باعث آسان‌تر کردن تخمین زدن کارها برای تیم است. تیم‌ها به‌جای نگاه کردن به product backlog و تخمین یکی از فقرات آن بر حسب ساعت، فقط در نظر می‌گیرند که این فقره به چه مقدار تلاش -نسبت به سایر فقرات product backlog- نیاز دارد. دلیل استفاده نکردن از معیارهای زمانی مختلف مانند ساعت یا دقیقه این است که افراد مختلف کارهای یکسان را زمان‌های متفاوتی انجام می‌دهند. پس با استفاده از story point سختی هر کار به صورت نسبی با سایر کارها محاسبه می‌شود و هر developer می‌تواند درک مناسبی از سختی کارها بدست آورد. این به معنای دقت بیشتر در تخمین است و در نهایت منجر به انتشار روان‌تر می‌شود. برای تخمین story point، به هر user story یک مقدار امتیاز اختصاص می دهیم. user story ای که ۲ story point به آن اختصاص داده شده باید دو برابر سخت‌تر از user story باشد که ۱ story point به آن اختصاص داده شده است. به محض انجام اولین اسپرینت، می‌دانیم که یک تیم می‌تواند در هر اسپرینت چند story point را تکمیل کند. پس با این روش متوجه می‌شویم که برای تکمیل پروژه به چند اسپرینت نیاز داریم و چگونه story point را به خطوط زمانی واقعی تبدیل کنیم.
تخمین story point بر اساس سه مؤلفه اصلی است:

1. ریسک شامل خواسته های مبهم، وابستگی‌ها و تغییرات تصادفی.
2. پیچیدگی مربوط به توسعه و پیاده‌سازی هر ویژگی.
3. تکرار که بر اساس میزان شناخت اعضای تیم از یک ویژگی -و پیاده‌سازی آن- و میزان یکنواخت بودن وظایف تعیین می‌شود.

در نهایت برای انجام تخمین story point ها هر تیم باید یک story پایه پیدا کند. این story لزوماً کم زمان‌ترین نیست، بلکه آن چیزی است که برای همه افراد تیم قابل درک است

و نشان دهنده story ای است که کمترین ریسک، پیچیدگی و تکرار را دارد (می‌توان به آن مقدار پایه ۱ را اختصاص داد). پس از تعیین آن، story point تمام user story باید با مقایسه آن‌ها تعیین شوند. هنگامی که تیم توسعه تخمین را انجام می‌دهد، توصیه می‌شود از روش‌های سنتی برای تعیین story point ها استفاده نشود. بلکه با استفاده از اعداد فیبوناچی (۱، ۲، ۳، ۵، ۸، ۱۳، ۲۱) این کار را انجام داد. مقیاس فیبوناچی چابک به تیم‌ها تخمین واقعی‌تری برای story point ها می‌دهند زیرا معمولا تشخیص تفاوت اعداد تخمینی بسیار نزدیک به هم بسیار سخت‌تر از تشخیص تفاوت میان اعداد با فاصله است. در نهایت، باید به این نکته توجه کرد که وقتی story point یک user story بیشتر از ۲۱ باشد، بهتر است که user story باید دوباره تقسیم شود.

سؤال ۴

چه فاکتورهایی در اولویتبندی آیتمهای Product backlog تاثیرگذار هستند؟

- رضایت مشتری
- ارزش های سازمان
  - هدف سازمان
  - استراتژی سازمان
- تعداد کاربر های تحت تاثیر قرار گرفته
  - برای مثال اگر نیاز باشد تغییرات ضروری روی وب سایت ایجاد کنیم. تغییراتی که روی صفحه ی اصلی سایت هستند از اهمیت بیشتری برخوردارند زیرا کاربران بیشتری درگیر آنها هستند.
- هزینه ی پیاده سازی
- ریسک های پیاده سازی
- پیچیدگی پیاده سازی

به چه روشهایی می توان آیتم ها را اولویت بندی کرد؟

مدل Stack Ranking

در این روش ابتدا بر اساس موارد قسمت قبل به یک ترتیب از اولویت ها برای backlog ها میرسیم سپس در گام اول کاری که بیشترین اولویت را دارد را انتخاب میکنیم سپس کار دوم را براساس کار اول و کار سوم را براساس دو کار قبلی و به همین شکل تسک ها را انتخاب میکنیم تا تسک ها تمام شوند.

مدل Kano

در این مدل تسک ها را به در 5 دسته ی زیر قرار می دهیم.

- دسته ی Must-Be

در این دسته تسک هایی قرار میگیرند که انجام شدنشان ضروریست و کابر انتظار دارد همچین قابلیت هایی وجود داشته باشد.

- دسته ی Attractive

تسک هایی که انجام شدنشان باعث خوشحالی کاربر میشود ولی انجام نشدن آن ها کاربر را ناراحت نمیکند.

- دسته ی One-Dimensional

تسک هایی که کاربر ها را با انجام شدنشان خوشحال و با انجام نشدنشان ناراحت می کند.

- دسته ی Indifferent

تسک هایی که تاثیری در رضایت کاربران ندارد. مانند ریفکتور کردن کود ها

- دسته ی Reverse

دسته ای که انجام شدنشان کاربر را ناراحت و انجام نشدنشان کاربر را خوشحال میکند. مانند اضافه کردن یک مرحله ی ورود اضافه برای افزایش امنیت که کاربران ساده را ناراحت میکند.
در ادامه با توجه به موقعیت از دسته های مختلف میتوان تسک ها را انتخاب کرد. در ابتدا به نظر میرسد باید صرفا تسک های Must-Be انجام داد اما برای رسیدن به موفقیت بالاتر بهتر است به تسک های Attractive و One-Dimensional نیز توجه کرد اما توصیه میشود با توجه به زمان و شرایط از هر کدام از دسته های بالا دسته کم یک تسک انتخاب شود.

مدل MoSCow

خلاصه ی اسم این مدل از عبارت های Must have, Should have, Could have, Won't have ساخته شده است و همانند مدل Kano، تسک ها را به این دسته ها اختصاص میدهد.
تفاوت این مدل با مدل Kano با این مدل در این است که مدل Kano رویکردی بازار محور اما این مدل رویکردی محصول محور دارد.

- دسته ی Must have

دسته ای که اگر تسک های آن را انجام ندهیم نمی توانیم محصول را عرضه کنیم. این دسته معادل با دسته ی Must-be در مدل Kano نیست و تسک های ضروری تری را شامل میشود. اما میتوان با ترکیب این مدل با مدل Kano، فرضا دسته های Must-Be, Attractive, One-Dimensional را در این دسته قرار داد تا ترکیبی از تسک های به اصطلاح user-friendly, market-ready را پوشش دهیم.

- دسته ی Should have

این دسته شامل تسک هایی میشوند که با وجود مهم بودن انجام دادن یا ندادنشان تاثیر مستقیم روی موفقیت محصول ندارند.

- دسته ی Could have

تسک های این دسته با وجود نیاز، اهمیت کمتری نسبت به دسته ی Should have دارند و انجام ندادن آن ها تاثیر کمتری در موفقیت محصول دارد.

- دسته ی Won't have

این دسته شامل تسک هایی میشود که تقریبا همه ی افراد با انجام ندادن آن ها در این زمان موافقت دارند. این تسک ها در backlog باقی می مانند تا زمان مناسب برای پیاده سازی آن ها فرا رسد.

سؤال ۵

یک داستان کاربر خوب در سه مرحله توصیف می‌شود:

- شرح خلاصه‌ای از نیازها
- مکالمه‌ای که در طول فرآیند اصلاح و پیشرفت برای اضافه کردن جزئیات صورت می‌گیرد
- آزمون‌هایی که برای تایید رضایت داستان انجام می‌گیرد

یک داستان خوب باید کیفیت‌هایش با استاندارد INVEST تطابق داشته باشد:

(I)ndependent:

داستان باید مستقل باشد و تحت تاثیر کسی یا چیزی قرار نگرفته باشد.

(N)egotiable:

تنها به نیاز های کاربر بپردازد، و باید توانایی بحث کردن داشته باشد. داستان کاربر نباید به شکل قرارداد نوشته شود

(V)aluable:

باید برای دیگر کاربران ارزشمند.

(E)stimable:

داستان کاربر باید قابلیت تخمین زدن را به مسئولان پروژه بدهد تا بتوانند تسک‌ها را آماده‌کرده تقسیم‌بندی کنند.

(S)mall:

داستان کاربر باید کوتاه باشد و در ۳-۴ روز تمام شود.

(T)estable:

باید تعدادی معیار از پیش‌نوشته وجود داشته باشد و داستان کاربر آن معیارها را در نظر داشته باشد.

بخش‌های مختلف User Story به فرم جملاتی هست که یک «نقش» یک «کار» را برای رساندن یک «فایده» به شرکت انجام می‌دهد.

بخش بعدی بحث و گفت‌و گو است که شامل مباحثه‌ای است که بین کاربران، تیم توسعه‌دهنده، و سهامداران شکل می‌گیرد. مواردی که باید پوشش داده‌شوند به صورت زیر است:

- مکالمات مشترک توسط سهامداران و تیم، برنامه ریزی میشود
- مکالمه در مورد ارزش داستان صورت میگیرد و برگه نوشته شده باید طوری تنظیم شود که به درک بهتر مکالمه کمک کند.
- مکالمه در اکثر مواقع به صورت شفاهی صورت میگیرد اما گاهی ثبت هم میشود تا مورد آزمایش قرار گیرد.

در مرحله بعدی باید تایید صورت بگیرد. مشتری یا مالک باید تایید کنند که داستان رضایت آن‌ها را جلب کرده‌است:

- مشتری یا مالک محصول باید تایید کند که داستان کاملا به اهداف خود رسیده است
- تیم توسعه و مشتری باید طبق تعریف و قوانین شرکت، پروژه را تکمیل کنند.
- برای بعضی داستان ها میشود معیارهای قبولی خاصی در نظر گرفت تا آنها هم تکمیل شوند، اما معیار های اصلی باید به خوب توسط تیم توسعه درک شوند و روی آنها توافق وجود داشته باشد. یک داستان باید تمام تست ها را با موفقیت پشت سر بگذارد.

بر طبق همه موارد اشاره شده به مثال زیر توجه کنید:

Title: Storing records
Priority: High
As a storage maintainer, I want to record buy and sell records so that the company can have statistics on the financial savings and increase the benefits.
Acceptance Criteria:
Given a full month of storage operation
A full record in a standard excel format provided should be recorded and a full analytic of the storage should be written for each month.